DENSITY ESTIMATION TECHNIQUES FOR GLOBAL ILLUMINATION

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Bruce Jonathan Walter

August 1998

DENSITY ESTIMATION TECHNIQUES FOR GLOBAL ILLUMINATION

Bruce Jonathan Walter, Ph.D.

Cornell University 1998

In this thesis we present the density estimation framework for computing view-independent global illumination solutions. The framework consists of three phases: particle tracing, density estimation, and decimation. Monte Carlo particle tracing is used to accurately simulate the light transport under a general spectral geometric-optics based physical model. Next kernel density estimation is used to reconstruct perceptual illumination functions. Finally decimation is used to optimize the resulting mesh for compactness and rapid interactive display as Gouraud-shaded triangles.

The three principal contributions of this work are the framework's separation of transport and function reconstruction computations, its ability to produce accurate solutions with precisely known error characteristics, and the techniques that we introduce to improve its efficiency and accuracy.

Particle tracing's generality allows us to eliminate or delay many common simplifying assumptions and improves our accuracy and error analysis. Delaying the density estimation until particle tracing is complete allows us to make better use of the expensive particle data. The separation of global transport and local representation computations also reduces the computational complexity of each phase, enhances the framework's scalability, and exposes abundant opportunities for parallelism. Another advantage is that we can solve directly for the radiant exitance without needing to estimate the more complicated spectral radiance function.

Despite its advantages, if naively implemented the framework would be prohibitively expensive. Thus we also introduce several techniques that significantly improve its accuracy and efficiency. These include the separation of luminance and chromaticity bandwidths, perceptually-motivated noise visibility predictors, statistical bias detection techniques to automatically enhance underresolved illumination features, a local polynomial density estimation method to eliminate boundary bias, and wavelength importance sampling to reduce the spectral noise. Results of the framework are shown for some complex environments and compared against measured data for a simple scene.

The strength of our framework is that it can simulate a wider variety of lighting effects, with fewer simplifying assumptions, and more precise error analysis that current view-independent methods. Furthermore, because of its accuracy, our density estimation framework solutions are used as reference solutions for judging the quality and effectiveness of more approximate but faster rendering methods.

Biographical Sketch

The author was born on May 10, 1969 and raised in Huntington, New York. Fleeing Long Island and the scourge of its dreaded accent, he attended Williams College in western Massachusetts. While there, he completed a BȦin 1991 with a double major in Physics and Computer Science thus causing even his physics advisor to complain that he wasn't taking enough liberal arts courses. Along the way he finished a undergraduate thesis on simulating hydrogen masers and wrote his first ray tracer.

As computer graphics seemed more fun than quantum physics, he started as a Ph. D. student in Computer Graphics at Cornell University. However, eighteen years of continuous schooling proved to be more than he could take and he soon left on a leave of absence. During this time he worked for the Caligari Corporation in California where he aquired the mystifying title of "Project Engineer" and shepherded the trueSpace 1.0 project into a successful shipping software product. After this two year stint in industry, much to everyone's surprise, perhaps including his own, he returned to finish his doctorate. Having received an offer he could not refuse, he plans to work as a post doctoral researcher in the IMAGIS/INRIA lab in Grenoble, France and contemplate a life as a permanent wandering post-doc.

Acknowledgements

I would like to thank my advisor Don Greenberg for his advice and help as well as for providing an wonderful environment for research. My minor advisor David Ruppert was an invaluable help in educating me on density estimation. Also thanks to my third committee member Thorsten von Eicken for his patience and insights on parallel processing.

Special thanks go to Peter Shirley for serving as my friend and mentor during his two year stint as a visiting faculty at Cornell and for converting me to the Monte Carlo faith. The density estimation framework originated in one of our many fascinating discussions. I also need to thank the entire density estimation project group. Pete for coordinating and implementing the particle tracer, Philip Hubbard for the decimator, Sebastian Fernandez for the walkthrough program and selflessly taking on demo duty, and Bretton Wade and Dave Zareski for their work on the initial implementation.

I also owe a debt of gratitude to the modelers, especially Chris Williams for the Library model, Albert Dicruttalo for the Science Center and Sculpture Gallery models, Modesto Bigas for the Kidosaki model, and Matt Hyatt and Stephen Holley for the Fallingwater model. Their models provided the challenging test cases that made the work exciting and provided the impetus to make the implementation ever more general and robust. Also thanks to Ben Trumbore for making the models available and even translating them into my prefered idiosyncratic format.

I would also like to thank the virtual lights team for making that project a success and joy and for padding my résumé. Thanks to Gün Alppay for doing all the hard work, Eric Lafortune for his ever helpful advice, and Sebastian for making the video possible.

Hurf Sheldon, Mitch Collinsworth, Martin Bergren, and the rest of the support staff made sure that the equipment I needed was (almost) always available and working and even procured a fast PC for me on short notice. Thanks to Linda Stephenson, Jonathan Corson-Rikert, Ellen French and Peggy Anderson for their every cheerful and helpful administrative support.

My thanks to all my fellow students at the lab without whose support and distractions, my stay would not have been nearly as enjoyable or educational. Particular thanks to Steve Marschner, Jed Lengyel, and Liang Peng for being great officemates and many intellectually challenging discussions. Thanks to Dan Kartch for maintaining Latex tools and James Durkin for maintaining the GNU tools.

Also special thanks to my undergrad advisor Don House who first introduced the field of computer graphics to me and got me hooked. Finally I'd l ike to thank my parents without whose continuous support none of this would have been possible. And last but not least, thanks to Geri for picking me up during my stressed out days and for making me sorry to leave Ithaca.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

This thesis presents a density estimation framework for computing view-independent global illumation solutions. We will discuss all the aspects of the framework including the particle tracing, density estimation, and decimation phases with special emphasis on the density estimation. Portions of this work have previously been published in in [47, 56], but several techniques including the automatic adaptive bandwidth selector and the wavelength importance sampling are presented here for the first time.

The goal of global illumination algorithms is to simulate the flow of light in an environment in a way that mimics the propagation of light in the real world. This is a complex problem and many different approaches have been proposed, but every approach represents a compromise among competing goals such as accuracy, interactivity, generality, and practicality. The true advantage of the density estimation framework is that it makes progress toward several of these goals simultaneously with particular emphasis on accuracy and generality.

The major contributions of this work are threefold. First is the framework itself and its total separation of transport and function estimation phases. These phases are computed using Monte Carlo particle tracing and kernel density estimation respectively. This separation reduces the computational complexity of each phase, enhances the framework's scalability, and exposes abundant potential parallelism.

The second contribution is the framework's accuracy and precisely characterized error. Particle tracing's generality allows us to eliminate or delay most of the common simplifying assumptions. It is also unbiased and simulates a general geometric-optics based model of light transport for increased accuracy. The density estimation introduces some bias but in a controlled and purely local fashion that is easy to analyze. Taken together, these components allow the density estimation framework to simulate a wider variety of lighting effects, with fewer simplifying assumptions, and a more precise error analysis than other current view-independent global illumination methods.

The third contribution is the efficiency and accuracy enhancing techniques that we introduce. If naively implemented, the framework would be too expensive to produce the highly accurate solutions that are its strength. Techniques such as our automatic adaptive kernel bandwidth selector make generating high quality solution feasible.

We begin this thesis with a brief description of our motivation and goals in designing a new global illumination algorithm. The physical and perceptual quantities that we are simulating are defined in Chapter 2 as well as a discussion of previous approaches and how they relate to our new framework. We give an overview of the density estimation framework in Chapter 3 and briefly describe each of its component phases: particle tracing, density estimation, and decimation. Next we describe the density estimation phase in greater detail. Chapter 4 covers kernel density estimation in general and our local polynomial approach to the problem of boundary bias. Chapter 5 describes the derivation of our automatic adaptive kernel bandwidth selector. The bandwidth is a crucial parameter in controlling solution quality and our approach includes both perceptually-motivated components and a statistical bias detection algorithm. Chapter 6 discusses some further issues and optimizations in implementing our framework including wavelength importance sampling. Results from our current implementation are presented in Chapter 7 including a comparision between a real measured environment and our simulation of it. Finally, Chapter 8 summarizes our findings.

1.1 Motivation

Realistic three dimensional graphics has enjoyed a tremendous amount of success and popularity in a wide variety of applications such as entertainment, sales, and visualization. It is no accident that these are also the uses with the least rigorous requirements for the global illumination. The results can be quite compelling but they need not be physically correct or predictive. There are many potential applications, particularly in design and simulation, that could be enabled if we had global illumination algorithms that were predictive and easy to use.

Perhaps the most obvious such application is in illumination engineering. A lighting engineer is responsible for ensuring that the lighting in an environment is both energy efficient and provides appropiate levels of light for the tasks to be performed there. Too much or too little light can cause discomfort, loss of productivity, and even injury. Their evaluation criteria can often be expressed as specific numerical ranges for physically measurable quantities such as candelas per square meters. A useful global illumination result would be to guarantee that the illuminance in a prospective environment configuration would be within the target range.

Another application with slightly different needs is architectural preview. An architect might be exploring a variety of possible building designs and want to evaluate their relative merits. In this case the evaluation is likely based on aesthetic as well as numerical criteria. Does the space have the kind of "feel" that the architect was trying to create. One crucial requirement for both of these applications is that the simulations must be predictive in order to be useful. The much easier goal of plausibility, which is necessary for many photorealistic applications, is not sufficient for these applications.

1.1.1 Goals

For realistic graphics to become truly useful as a predictive design tool, we will need to keep developing improved global illumination algorithms to better meet these users' needs. To help us in this pursuit, it is useful to list some of the goals that we would like to achieve. This list represents a continuing challenge as no current or forseeable algorithm will be able to perfectly achieve all of these goals simultaneously. Some of our goals are:

Predictive: If we really expect a designer to use lighting simulations to help choose among the variety of possible designs, then it is imperative that the simulations accurately predict the consequences of those choices. Moreover since some inaccuracy is inevitable, it is important to know and to communicate the expected accuracy of the results. Only then can the simulations be used as a rational basis for decision making.

Interactive: Ideally the simulation should provide instantaneous feedback whenever the user wants. The goal is to allow the user to explore the model and design spaces in a natural manner and at their own pace. In practice we will often have to settle for lesser forms of interactivity such as interactive walkthroughs of static environments.

Automatic: The process should seem like a "black box" to the users. They should not need to understand the inner workings of the global illumination algorithm or need to "tweak" its internal parameters. They may need controls to specify which information is of most interest, but any such controls should be high-level and intuitive.

General: The simulation should handle a broad class of possible inputs. In our case this means complex geometry, complex luminaires and complex material (BRDF) properties. The users' choice of designs should not be constrained by simulation software.

1.1.2 Why the Density Estimation Framework

What makes the density estimation framework interesting is that it makes progress toward several of these goals simultaneously.

In a realistically complex environment, the spectral radiance function, which describes the complete illumination in an environment, is generally too complicated to be solved for, stored, or computed with precisely. Unlike traditional finite element approaches, the density estimation framework is designed to work without simplifying assumptions or the need to find the spectral radiance explicitly. Instead the light transport is simulated using Monte Carlo particle tracing and then simplified functions that we can recover accurately are estimated from the particle statistics. This approach allows the framework to take a much more rigorous approach to accuracy and generality.

It also greatly simplifies our error analysis. The particle tracing is unbiased though noisy. The density estimation reduces this noise to an acceptable level by introducing bias, but does so in a controlled and purely local fashion that makes the bias easy to characterize. Together with the noise estimates we compute during the density estimation, this completely describes the error in our results and gives us much more precise error information than is available in other view-independent global illumination algorithms.

Interactivity and efficiency is perhaps the weakest attribute of the density estimation framework. Although we have made considerable improvements in its efficiency, it still takes many hours of computation to produce a high quality solution. Fortunately this can be reduced considerably by using parallel processing and many further algorithmic improvements remain to be made. Because we produce view-independent solutions, we are still able to achieve a limited form of interactivity. Once the solution has been computed, the user can wander freely throughout the environment at interactive rates as long as they do not change the enviroment. Such walkthroughs are quite valuable as they often reveal features and problems that are much harder to find by examining a few images from static viewpoints.

Another strength is that we have developed automatic techniques to choose the solution parameters based on perceptual and statistical measures. This is a hidden burden in many global illumination methods where correct user-provided algorithm-specific hints are required for the algorithm to work acceptably.

It is the combination of these strengths that makes the density estimation framework a significant contribution toward the goal of predictive global illumination. It is our hope that this work will stimulate further research toward this very challenging goal.

Chapter 2

Radiometry, Photometry, and Colorimetry

Before we can quantitatively discuss light and its flow through an environment, it is necessary to establish some basic definitions and notation. Wherever possible we follow the ANSI standard for illumination engineering nomenclature [30] as it represents the best current candidate for establishing a single standard notation in the graphics and global illumination communities.

2.1 Spectral Radiometry

The most common measure of light used in the graphics literature is the *radiance*. However some caution is required as this term is often used ambiguously. The term *radiance* is frequently used to refer to any one of a family of related functions.

The fundamental member of this family is the *spectral radiance*, which is denoted L_λ. The spectral radiance is a function of four variables: position, direction, wavelength, and time. The amount of energy flowing through a position $\mathbf{x}$, in direction ω, as light of wavelength λ, at time t is written as $L_\lambda(\mathbf{x}, \omega, \lambda, t)$. Radiance is measured in units of

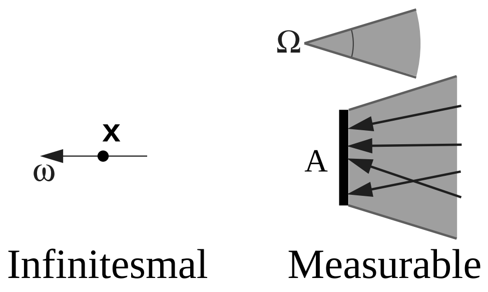

Figure 2.1: While the radiance is defined at each infinitesmal point, it is only physically meaningful when measured over a non-infinitesmal region. In order to collect a measurable amount of light energy, a physical sensor must be sensitive to light over a finite area A and finite solid angle Ω.

joules per meter squared per steradian per nanometer per second or $\frac{\text{watts}}{\text{m}^2\,\text{sr}\,\text{nm}}$ [1].

In theory, we could directly measure the spectral radiance by placing an appropiate sensor in the path of the light except that the spectral radiance at a point represents an infinitesmal amount of energy. In order to collect a measurable amount of energy, our sensor will have to collect light energy over some area A, some solid angle or range of directions Ω, some range of wavelengths, Λ, and some time interval, T. The amount of energy Q that our virtual sensor would collect is:

$$Q = \int_A \int_\Omega \int_\Lambda \int_T L_\lambda(\mathbf{x}, \omega, \lambda, t)\,[\omega \cdot \hat{\mathbf{n}}]\,dt\,d\lambda\,d\omega\,d\mathbf{x} \tag{2.1}$$

where $\hat{\mathbf{n}}$ is the surface normal of the sensor's collecting area. The dot product $[\omega \cdot \hat{\mathbf{n}}]$ is equal to the cosine of the angle between the direction ω and the surface normal $\hat{\mathbf{n}}$. This cosine factor is due to the fact that the sensor's effective collecting area, also called its projected area, decreases as the angle between the surface normal the light direction increases (Figure 2.2).

[1] A watt is a joule per second.

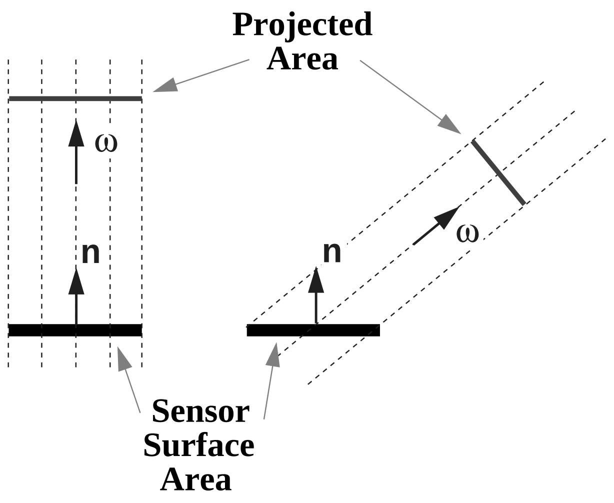

Figure 2.2: The effective collecting area, which is also called the projected area, of a sensor changes with angle. When the light direction ω is aligned with the surface normal $\hat{\mathbf{n}}$ (left), the sensor's projected area is the same as its surface area. However its projected area decreases as the angle between ω and $\hat{\mathbf{n}}$ increases (right). The projected area equal to the surface area times the cosine of the angle between the light direction ω and the surface normal $\hat{\mathbf{n}}$.

2.1.1 Sensor Specific Radiometry

Because the spectral radiance is a seven[2] dimensional function, it is often quite complex and difficult to estimate, manipulate, and store. When possible we would prefer to use lower dimensional functions that are simpler and easier to handle. One way to reduce the dimensionality is to specialize the radiance function for a particular type of sensor.

A sensor's response to light energy typically varies depending on the wavelength of the light. This dependence can be expressed as a spectral response function which we will write generically as $r(\lambda)$. We can create a specialized *radiance* function as:

$$L(\mathbf{x}, \omega, t) = \int_0^\infty L_\lambda(\mathbf{x}, \omega, \lambda, t)\, r(\lambda)\, d\lambda \tag{2.2}$$

which describes the light in terms of the potential response of our particular sensor. It no longer contains enough information to predict the response of other sensors.

The exact spectral response function used will depend on the application. Heat transfer applications are interested in how much energy reaches a surface regardless of its wavelength and the appropiate spectral response function in this case is $r(\lambda) = 1$. This particular specialized radiance function is denoted L_e and called the radiance. In monochrome imaging applications, we are interested in the brightness to a human observer and the most appropiate spectral response function is the CIE Standard Luminous Efficiency Function $V(\lambda)$ [61]. In this case the specialized radiance is denoted L_v and is called the *luminance*. The various radiance functions generally have distinguishing names and subscripts, however it is common practice to leave off the subscript and simply refer to the application-specific function as the radiance[3]. In this thesis we will specify the exact function we mean and only use the unqualified "radiance" terminology with statements that are true for all radiance functions.

[2]Three spatial dimensions, two directional dimensions, one spectral dimension and one dimension for time

[3]Although the official definition of the term radiance is L_e [30], this is rarely, if ever, what is meant when the term radiance is used in the computer graphics community.

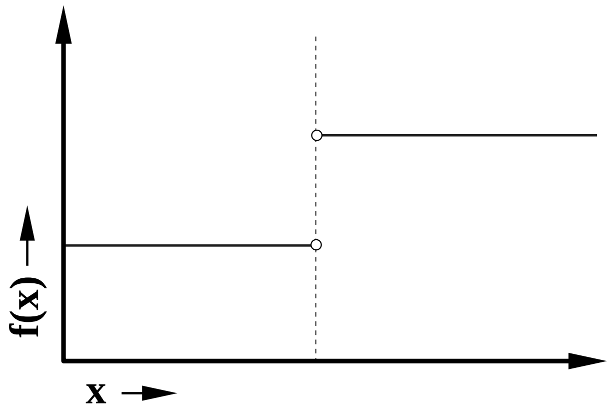

Figure 2.3: A jump disconuity in one dimension. The function is well defined on either side of the discontinuity, but the value exactly at the discontinuity can be defined arbitrarily. The open circles are the two choices that come from enforcing continuity from one side or the other, known as left or right continuity. We resolve this ambiguity for the radiance at surfaces using a similar constraint which we call forward continuity.

2.1.2 Radiometry at Surfaces

Another problem that we need to clarify is the definition of the radiance at surfaces. In the graphics community, we generally model our surfaces as being infinitely thin or two dimensional (e.g., polygons or spline patches). As a result the radiance along a ray will change discontinuously when it crosses a surface. The radiance is well defined on either side of the surface but its not obvious what value it should have exactly at the surface. Figure 2.3 shows this ambiguity for a 1D function. The two most reasonable choices are to define it by continuity from one side or the other. In essence our choice is, should the radiance at a surface be defined as the amount of light reaching the surface or the amount of light leaving the surface. We are free to choose either, but we want our definition of radiance to be unambiguous and mathematically well defined.

We resolve this ambiguity by defining the radiance to be forward continuous. Let us quickly introduce some notation to make it clear what this means. We represent directions by the symbol, ω. Given a point $\mathbf{x}$ then we define $\mathbf{y} = \mathbf{x} + a\omega$ to be the point that we would arrive at by starting at $\mathbf{x}$ and then moving a distance a in direction ω. We also define $-\omega$ to be the direction opposite to ω. Thus we can return to point $\mathbf{x}$ by starting at $\mathbf{y}$ and moving distance a in direction $-\omega$ (i.e. $\mathbf{x} = \mathbf{y} + a(-\omega) = \mathbf{y} - a\omega$). We can now write our forward continuity condition as:

$$L(\mathbf{x}, \omega, \lambda, t) = \lim_{a \downarrow 0} L(\mathbf{x} + a\omega, \omega, \lambda, t) \tag{2.3}$$

Note that this is a one-sided limit as the strictly positive a decreases toward zero. With this constraint, $L(\mathbf{x}, \omega, \lambda, t)$ will always be continuous in direction ω. There is still an ambiguity about the radiance at a point on a planar surface in a direction parallel to the surface, but this will not effect our computations and we are free to choose the radiance to be zero in this case.

Since we defined the radiance as the light energy traveling in direction ω, our continuity constraint means that the value of the radiance function at a surface is equal the outgoing radiance (i.e. the light leaving the surface). It is also useful to have a convenient way to refer to the incoming radiance at a surface. We can define a function L_{i} called the incoming radiance that measures the amount of light coming *from* a direction[4]. Compared to the usual radiance, this function reverses the meaning of its directional argument. When we want to emphaize this distinction, we can relabel the usual radiance as the outgoing radiance L_{o}. The incoming radiance still obeys Equation 2.3, but its value at surfaces is the incoming light rather than the outgoing light. If there is not a discontinuity in the radiance function at $\mathbf{x}$ (i.e. $\mathbf{x}$ is not on a surface), then $L_{\mathrm{i}}(\mathbf{x}, \omega, \lambda, t) = L_{\mathrm{o}}(\mathbf{x}, -\omega, \lambda, t)$, and in all cases we have:

$$L_{\mathrm{i}}(\mathbf{x}, \omega, \lambda, t) = \lim_{a \downarrow 0} L_{\mathrm{o}}(\mathbf{x} + a\omega, -\omega, \lambda, t) \tag{2.4}$$

Some authors choose to only define the radiance at surfaces, but there is no real reason not

[4]An alternate notational convention (e.g., in [20]) is to overload the directional argument to also include a continuity constraint. For each physical direction then there are two distinct directions whose only difference is that they imply a different continuity constraint. This seems counter-intuitive and confusing to us.

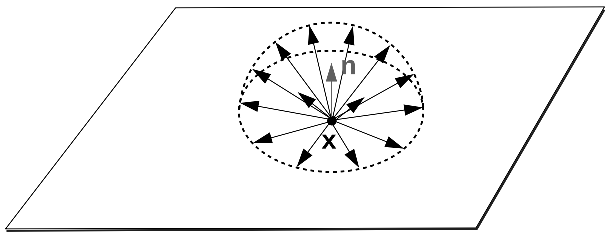

Figure 2.4: The hemisphere of directions $\Omega_{\hat{\mathbf{n}}}$ about a surface normal.

to define it over all space. A full definition gives a clean conceptual basis for extensions such as transmissive surfaces and participating media.

2.1.3 Irradiance and Radiant Exitance

Now that we can refer unambiguously to the radiance at surfaces, we can define some further quantities that are useful at surfaces. The *spectral irradiance* E_λ measures how much light energy is striking one side[5] of a surface from all directions and is defined by:

$$E_\lambda(\mathbf{x}, \lambda, t) = \int_{\Omega_{\hat{\mathbf{n}}}} L_\mathrm{i}(\mathbf{x}, \omega, \lambda, t)[\omega \cdot \hat{\mathbf{n}}] \, d\omega \tag{2.5}$$

where again $\hat{\mathbf{n}}$ is the surface normal at the point $\mathbf{x}$ and $\Omega_{\hat{\mathbf{n}}}$ is the hemisphere of directions centered about the surface normal. It will hopefully be clear which of the two possible surface normal directions is meant and we will not usually bother to specify this explicitly. Since it is defined in terms of the L_i, the irradiance measures the amount of light striking the surface. There is also an analogous quantity for the light leaving one side of a surface called the *spectral radiant exitance* M_λ and defined by:

$$M_\lambda(\mathbf{x}, \lambda, t) = \int_{\Omega_{\hat{\mathbf{n}}}} L_\mathrm{o}(\mathbf{x}, \omega, \lambda, t)[\omega \cdot \hat{\mathbf{n}}] \, d\omega \tag{2.6}$$

[5]Note there is some ambiguity as to whether the irradiance and radiant exitance refer to the light on one side of a surface or both sides. We have chosen to define it as one-sided as this seems more useful.

These functions can be specialized for particular spectral response functions and are then denoted using the same subscripts as with the radiance. Many other specialized radiometric function have been defined (e.g., see [30, 20]) which will not be considered here. An alternate derivation of radiometric measures which readers may find helpful is given by Arvo [5] and is explicitly based on a particle model of light.

2.2 Colorimetry and Photometry

In realistic computer graphics we are usually interested in the appearance of an environment to a human observer. We can use this to our advantage because humans see colors rather than spectra. Quite different spectra can be perceived as being the same color and hence indistinguishable to an observer. Such spectra are known as *metamers*, and their existence implies that we actually need significantly less information than if we had to reconstruct complete spectra.

For an normal observer in the range of ordinary color vision, also called the photopic region, the *trichromatic generalization* [15, 61] has become well established. The idea is that any color stimulus can be duplicated by an appropiate combination of three primary colors. The coefficients which describe this mixture are known as the color's *tristimulus values*. The actual numerical values depend on which particular colors we chose as our primaries, but the important point is that we can adequately represent any color stimulus with just three numbers.

The most widely used standard tristimulus space is the *CIE 1931 Standard Colormetric Observer* [61, 20, 30]. This tristimulus space consists of three channels called X, Y, and Z and defined by their corresponding spectral response functions: $\bar{x}(\lambda)$, $\bar{y}(\lambda)$, and $\bar{z}(\lambda)$ which are shown in Figure 2.5. The XYZ primaries are imaginary in that individually they are not physically realizable color stimuli, but there is no mathematical difficulty in representing all real color stimuli as a combinations of these imaginary primaries.

The CIE XYZ space and response functions provides us with a standard and well

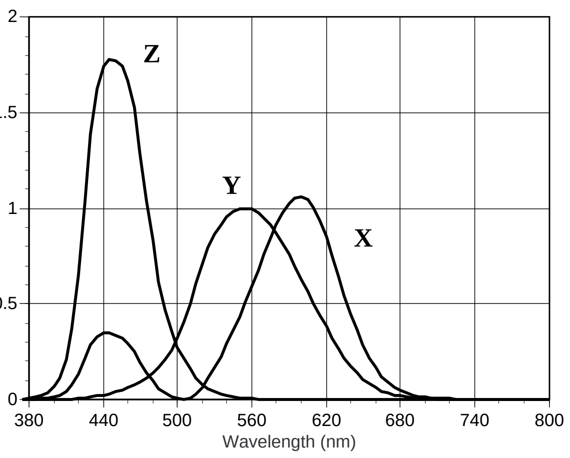

Figure 2.5: The XYZ spectral response functions. The X, Y, and Z tristimulus values of a stimulus are found by integrating its spectrum against the corresponding spectral response functions $\bar{x}$, $\bar{y}$, and $\bar{z}$ which are shown above.

defined way to convert from a physical/spectral space to a perceptual/tristimulus space. Given a color stimulus with spectrum $s(\lambda)$, we can compute its XYZ tristimulus values using:

$$\mathrm{X} = 683 \int_0^\infty s(\lambda)\,\bar{x}(\lambda)\,d\lambda \tag{2.7}$$

$$\mathrm{Y} = 683 \int_0^\infty s(\lambda)\,\bar{y}(\lambda)\,d\lambda \tag{2.8}$$

$$\mathrm{Z} = 683 \int_0^\infty s(\lambda)\,\bar{z}(\lambda)\,d\lambda \tag{2.9}$$

where 683 is a required conversion constant.

The XYZ primaries were chosen so that the XYZ tristimulus values of any physical stimulus are always positive and $\bar{y}(\lambda) = V(\lambda)$. As mentioned earlier, $V(\lambda)$ is the luminous efficiency function standardized by CIE in 1924. It is intended to correspond roughly to the human perception of brightness. Thus the Y value gives us the brightness of a color. It also useful to have a way to refer to the parts of a color stimulus that are "orthogonal" to the brightness (i.e. invariant under a simple change in overall intensity) such as hue and saturation. Chromaticity coordinates, such as x and y, are frequently used for this purpose.

$$x = \frac{\mathrm{X}}{\mathrm{X}+\mathrm{Y}+\mathrm{Z}} \tag{2.10}$$

$$y = \frac{\mathrm{Y}}{\mathrm{X}+\mathrm{Y}+\mathrm{Z}} \tag{2.11}$$

Together these are known as a stimulus' chromaticity. It is easy to convert between a color's Yxy coordinates and its XYZ tristimulus values.

2.2.1 Photometry and Tristimulance

When the luminous efficiency function is used as the spectral response function to create specialized radiometric functions (as in Equation 2.2), these functions have special names and are known collectively as photometric functions and denoted with the subscript v. The luminance L_v is derived from the spectral radiance L_λ by:

$$L_\mathrm{v}(\mathbf{x},\omega,t) = 683 \int_0^\infty L_\lambda(\mathbf{x},\omega,\lambda,t)\, V(\lambda)\, d\lambda \tag{2.12}$$

where 683 lumens/watt is a required conversion constant. The illuminance E_v and luminous exitance M_v are derived similarly from the spectral irradiance E_λ and the spectral radiant exitance M_λ respectively. Thus they measure the brightness of the light striking and leaving a surface respectively.

Photometry is sufficient for monochromatic applications, but for color applications we need all three XYZ channels. The quantity we ultimately need is the "tristimulus values of the spectral radiance". Unfortunately, there does not seem to be an standard name for this quantity. One can avoid this lack of a concise name by refering to this quantity as luminance plus chromaticity (i.e. Yxy coordinates). We consider this terminology to be a little awkward because it does not correspond how we actually compute or store it. Instead in this thesis we have invented names[6] Although these names are non-standard, their use will ease our discussion somewhat.

[6]The three terms propose are the tristimulance, intristimulance and tristimulus exitance for the tristimulus values of the spectral radiance, spectral irradiance, and spectral

The tristimulus values of the spectral radiance, we term the *tristimulance*, denoted by L_t. When expressed in terms of CIE XYZ space the components of L_t will be denoted $L_{\mathrm{t},X}$, $L_{\mathrm{t},Y}$, and $L_{\mathrm{t},Z}$ and defined by:

$$\begin{aligned} L_{\mathrm{t},X}(\mathbf{x},\omega,t) &= 683\int_0^\infty L_\lambda(\mathbf{x},\omega,\lambda,t)\,\bar{x}(\lambda)\,d\lambda \\ L_{\mathrm{t},Y}(\mathbf{x},\omega,t) &= 683\int_0^\infty L_\lambda(\mathbf{x},\omega,\lambda,t)\,\bar{y}(\lambda)\,d\lambda \\ L_{\mathrm{t},Z}(\mathbf{x},\omega,t) &= 683\int_0^\infty L_\lambda(\mathbf{x},\omega,\lambda,t)\,\bar{z}(\lambda)\,d\lambda \end{aligned} \tag{2.13}$$

Earlier we introduced the spectral irradiance E_λ and spectral radiant exitance M_λ as being useful measures of the aggregate incoming and outgoing light at a point on a surface. We define tristimulus versions of these functions as the *intristimulance*, E_t, and *tristimulance exitance*, M_t. Much of this thesis will be devoted to computing the tristimulus exitance which from Equations 2.6 and 2.13 is defined by:

$$\begin{aligned} M_{\mathrm{t},X}(\mathbf{x},t) &= 683\int_{\Omega_{\hat{\mathbf{n}}}}\int_0^\infty L_\lambda(\mathbf{x},\omega,\lambda,t)\,\bar{x}(\lambda)\,d\lambda\,[\omega\cdot\hat{\mathbf{n}}]\,d\omega \\ M_{\mathrm{t},Y}(\mathbf{x},t) &= 683\int_{\Omega_{\hat{\mathbf{n}}}}\int_0^\infty L_\lambda(\mathbf{x},\omega,\lambda,t)\,\bar{y}(\lambda)\,d\lambda\,[\omega\cdot\hat{\mathbf{n}}]\,d\omega \\ M_{\mathrm{t},Z}(\mathbf{x},t) &= 683\int_{\Omega_{\hat{\mathbf{n}}}}\int_0^\infty L_\lambda(\mathbf{x},\omega,\lambda,t)\,\bar{z}(\lambda)\,d\lambda\,[\omega\cdot\hat{\mathbf{n}}]\,d\omega \end{aligned} \tag{2.14}$$

2.2.2 Other Color Spaces

While XYZ is our standard color space, there are occasionally good reasons to work with other color spaces. The two examples we will mention here are an opponent-color space and CIE L*a*b* space. We will have more to say about these color spaces in Chapter 5.

The choice of the XYZ primaries was largely arbitrary; one can equivalently use any linear transform of this space. There is good evidence [43, 28] that, at least at some point in our visual system, color stimuli are actually encoded as a luminance channel

radiant exitance respectively. While these terms are non-standard, we consider these quantities important enough to be given names and hope the graphics community will eventually adopt standard names for them.

plus two color-opponent channels which correspond roughly to the general categories: brightness, red vs. green, and yellow vs. blue. Experiments [43] have shown that these channels have differing sensitivities and are approximately separable. The ability to treat these channels as being roughly independent can be a strong advantage and is not true of XYZ space. The use of color-opponent channels allows to solve for individual channels independently with much less worry about problems due to correlations and covariances between channels.

We often want to measure how perceptually different two colors are. In XYZ space, this is not straightforward as the distance[7] between colors does not correspond very well with their perceptual difference. A perceptually uniform space is one where Euclidean distance and perceptual difference would be the same, and a variety of color spaces have been designed with this as their goal. CIE L*a*b* space is currently the best of these uniform spaces [15]. The transform from XYZ space to CIE L*a*b* space is given by:

$$\begin{aligned} \mathrm{L}^* &= 116\, g(\frac{\mathrm{Y}}{\mathrm{Y}_n}) - 16 && (2.15)\\ \mathrm{a}^* &= 500\left[g\left(\frac{\mathrm{X}}{\mathrm{X}_n}\right) - g\left(\frac{\mathrm{Y}}{\mathrm{Y}_n}\right)\right] && (2.16)\\ \mathrm{b}^* &= 200\left[g\left(\frac{\mathrm{Y}}{\mathrm{Y}_n}\right) - g\left(\frac{\mathrm{Z}}{\mathrm{Z}_n}\right)\right] && (2.17) \end{aligned}$$

$$g(x) = \begin{matrix} x^{1/3} & \text{if } x > 0.008856 \\ 7.787x + \frac{16}{116} & \text{if } x \leq 0.008856 \end{matrix} \qquad (2.18)$$

In 1994 CIE further updated this space by adopting a distance measure that differs slightly from Euclidean distance[15, p$\dot{9}$4]. The complete procedure for estimating perceptual color difference is to transform the colors into CIE L*a*b* space and then apply the modified distance formula.

[7]Distance here means the usual or Euclidean distance between values when viewed as points in 3D space.

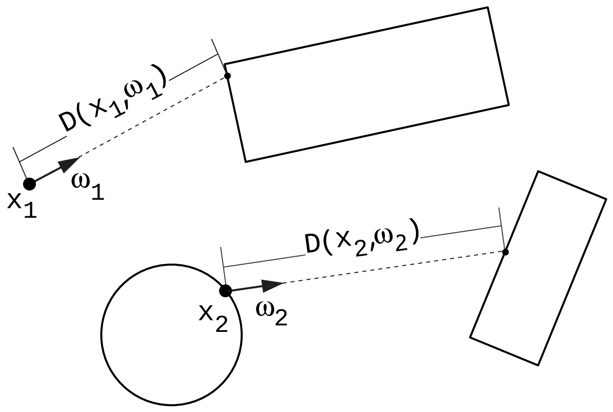

Figure 2.6: The distance function $D(\mathbf{x}, \omega)$ returns the distance to the first intersection of the corresponding ray with the surfaces in the environment.

2.3 Light Transport

Now that we have defined the measures to talk quantitatively about light, next we need to formulate the equations that will govern our simulation of the flow of light throughout an environment. The first decision is at what level should we model the light. Following the general consensus of the global illumination community, we use a geometric optics based approximation. This approximation includes most of the visually important lighting phenomena, but it does exclude long range wave effects such as diffraction and interference.

The principle assumption is that light travels in straight lines between scattering events. For simplicity, let us assume that scattering only occurs at surfaces; this is known as the non-participating medium assumption. One consequence of the geometric optics approximation is that in the absence of scattering, the radiance of light remains constant as it travels along its ray [40]. In our case, this means its radiance can only change at surfaces.

We can formulate this precisely with the help of a few definitions. Let $\mathcal{M}$ be the set of surfaces in our environment and let $D(\mathbf{x}, \omega)$ be a distance function whose value is the

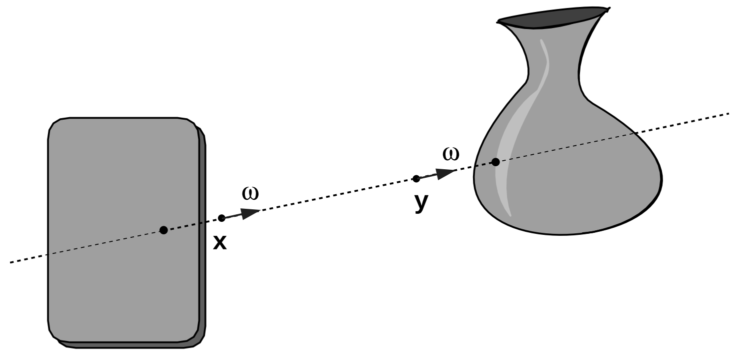

Figure 2.7: The radiance of light does not change as it travels unless something such as a surface scatters the light. Thus the radiance remains the same between points **x** and **y** but may change after it strikes the surface to the right.

distance to the nearest surface along the corresponding ray (Figure 2.6 and defined by:

$$D(\mathbf{x}, \omega) = \min_{a>0} \{a \,:\, \mathbf{x} + a\omega \in \mathcal{M}\} \tag{2.19}$$

We can state the constancy of radiance in the absence of scattering as:[8]

$$L_\lambda(\mathbf{x}, \omega, \lambda, t) = L_\lambda(\mathbf{x} + a\omega, \omega, \lambda, t + \frac{an}{c}) \quad \text{if } a < D(\mathbf{x}, \omega) \tag{2.20}$$

The radiance of the light does not change as it travels through time and space unless an obstruction scatters the light (Figure 2.7).

If we assume that the speed of light is extremely large compared to the distances and time scales of interest and neglect the propagation delays in the flow of light, we get:

$$L_\lambda(\mathbf{x}, \omega, \lambda, t) = L_\lambda(\mathbf{x} + a\omega, \omega, \lambda, t) \quad \text{if } a < D(\mathbf{x}, \omega) \tag{2.21}$$

This is equivalent to assuming that the speed of light is infinite, or alternatively that the flow of light is always in static equilibrium. In essence we assume that the lighting

[8]Note c is the speed of light in a vacuum (3x10^8 m/s), n is the index of refraction of the medium through which the light is traveling (e.g., $n \approx 1$ for air or $n \approx 1.4$ for glass), and $\frac{n}{c}$ is the speed of light in this medium.

at any point in time is the same as if conditions were always the same at they are at that moment. With this assumption the time parameter will always be a constant in our equations, and we will no longer include it explicitly.

2.3.1 BSDF

We still need some way to describe how the light scatters at surfaces or other obstructions. The standard method is with a *bidirectional scattering distribution function*, or BSDF, denoted by f_{s}. A BSDF describes how much of the light of wavelength λ which is incident at $\mathbf{x}$ location on a surface from direction ω with be scattered in direction ω' and is defined by:

$$f_{\mathrm{s}}(\mathbf{x}, \omega, \omega', \lambda) = \frac{L^*_{\mathrm{o}}(\mathbf{x}, \omega', \lambda)}{L_{\mathrm{i}}(\mathbf{x}, \omega, \lambda)\,|\omega \cdot \hat{\mathbf{n}}|} \tag{2.22}$$

We use L^*_{o} to indicate that this is the outgoing radiance induced by light from the specified incoming direction. It does not include self-emitted light or light scattered from other directions. The factor $|\omega \cdot \hat{\mathbf{n}}|$ is simply part of the official definition of the BSDF and is equal to the absolute value of the cosine the angle between the incoming light and the surface. Its rationale is the projected area effect we discussed earlier. We can find the total radiance being scattered from a surface in a particular direction by summing the contributions induced by all possible incoming directions:

$$L_{\mathrm{scat}}(\mathbf{x}, \omega', \lambda) = \int_{\Omega_\odot} f_{\mathrm{s}}(\mathbf{x}, \omega, \omega', \lambda,) L_{\mathrm{i}}(\mathbf{x}, \omega, \lambda)\,|\omega \cdot \hat{\mathbf{n}}|\; d\omega \tag{2.23}$$

where $\Omega_\odot$ is the sphere of all possible directions.

Note that the definition of the BSDF implicitly assumes that the light scattering is linear; how a part of the incident light is scattered does not depend on the total amount of incident light. Non-linear scattering effects do exist and can be important in some cases (e.g., self-darkening sunglasses or frequency doubling crystals), however we will assume that these are negligible in our environments.

For opaque surfaces, the BSDF is zero unless both directions lie on the same side of the surface. In this case the *bidirectional reflectance distribution function*, or BRDF,

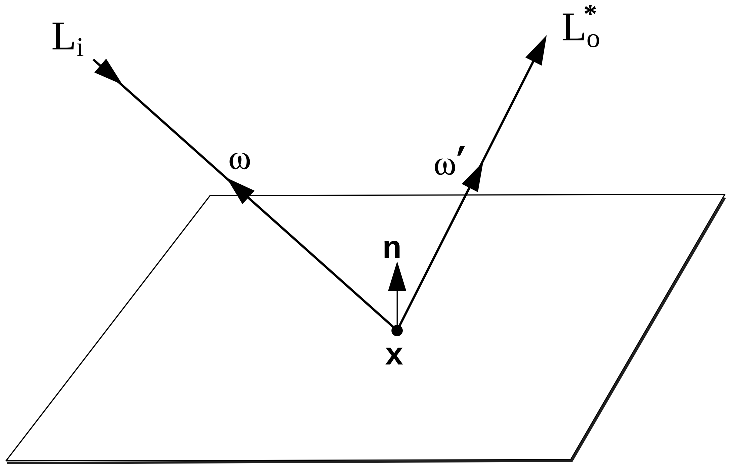

Figure 2.8: Of the light that is incident on a surface at a point $\mathbf{x}$ from direction ω the fraction that is scattered in direction ω' determines the BSDF (bidirectional scattering distribution function).

f_r can be used instead of the BSDF f_s. The only difference is that the BRDF is only defined when both incoming and scattering directions lie on the same side of the surface (i.e. it is a subset of the BSDF). Some care is required when creating a BSDF that includes transmission as well as reflection. Thus for simplicity, many authors only mention BRDFs. See [53] for a discussion of some of these issues.

BSDFs and BRDFs may come from many sources. They can be measured by a device called a gonioreflectometer (e.g., [22]), calculated from theoretical models (e.g., [26, 41, 13]), estimated by numerical simulations (e.g., [59]), or created using empirical representations (e.g., [58, 34]). If we want our results to be accurate, it is essential that our BSDFs be accurate. For our purposes, we assume that the BSDFs are part of our input and that they have been correctly specified.

Because of their simplifying properties, Lambertian BRDFs play a special role in many global illumination algorithms. A Lambertian surface is an opaque surface whose

BRDF does not depend on either the incoming or outgoing direction.

$$f_{\text{r,Lambertian}}(\mathbf{x}, \omega, \omega', \lambda) = \rho_{\mathbf{x}}(\lambda) \tag{2.24}$$

where ρ is known as the *spectral reflectance*. The major advantage of Lambertian surfaces is that their reflected radiance does not depend on the exiting angle. If the surface is not itself a light source then there is a direct mapping between the radiance L and radiant exitance M at a Lambertian surface:

$$L(\mathbf{x}, \omega, \lambda) = \frac{1}{\pi} M(\mathbf{x}, \lambda) \quad \text{if } \omega \in \Omega_{\hat{\mathbf{n}}} \tag{2.25}$$

The factor of π is because $\int_{\Omega_{\hat{\mathbf{n}}}} [\omega \cdot \hat{\mathbf{n}}] \, d\omega = \pi$. The same relation also holds between the tristimulance and the tristimulus exitance so that the color of a Lambertian surface does not depend on the direction from which you view it. This greatly simplifies the global illumination problem on such surfaces as we can solve for the radiant or tristimulus exitance rather than the full higher dimensional radiance function.

2.3.2 Global Balance Equation

Equation 2.23, the local scattering equation, can be extended into one expressing the total light flow in an environment. First we note that the radiance coming from a surface consists of two components: the light being emitted by the surface and the light being scattered by the surface. Thus we can write:

$$\begin{aligned} L(\mathbf{x}, \omega', \lambda) &= L_{\text{emit}}(\mathbf{x}, \omega', \lambda) + L_{\text{scat}}(\mathbf{x}, \omega', \lambda) \\ L(\mathbf{x}, \omega', \lambda) &= L_{\text{emit}}(\mathbf{x}, \omega', \lambda) + \int_{\Omega_{\odot}} f_{\text{s}}(\mathbf{x}, \omega, \omega', \lambda) L_{\text{i}}(\mathbf{x}, \omega, \lambda) \, |\omega \cdot \hat{\mathbf{n}}| \, d\omega \end{aligned} \tag{2.26}$$

Using the non-participating media assumption, the constancy of radiance along a ray implies that the incoming radiance at point $\mathbf{x}$ equal to the outgoing radiance at some other point $\mathbf{y}$ (Figure 2.9). Using Equation 2.19, we can define a geometry function, $G(\mathbf{x}, \omega)$ to give us this other point.

$$G(\mathbf{x}, \omega) = \mathbf{x} + D(\mathbf{x}, \omega) \, \omega \tag{2.27}$$

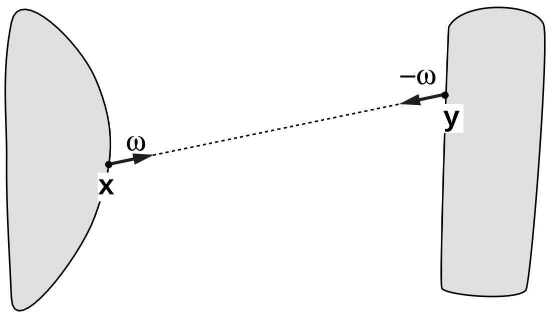

Figure 2.9: Assuming non-participating media, the incident radiance at a point on one surface is equal to the outgoing radiance a corresponding point on another surface. In the configuration shown above, $L_{\mathrm{i}}(\mathbf{x}, \omega, \lambda) = L_{\mathrm{o}}(\mathbf{y}, -\omega, \lambda)$.

We now complete our balance equation by making use of Equations 2.21 and 2.4 to get:

$$L_\lambda(\mathbf{x}, \omega', \lambda) = L_{\mathrm{emit}}(\mathbf{x}, \omega', \lambda) + \int_{\Omega_\odot} f_{\mathrm{s}}(\mathbf{x}, \omega, \omega', \lambda)\, L_\lambda(G(\mathbf{x}, \omega), -\omega, \lambda)\, |\omega \cdot \hat{\mathbf{n}}|\; d\omega \tag{2.28}$$

The balance equation completely specifies the radiance in our environment; only the correct radiance function will satisfy this equation. Note that because of our non-participating media assumption, we need only know the radiance at surfaces which can then be trivially extended to all space.

Equations of this general form were first introduced to the graphics community as the rendering equation [33, 29]. Since then several different but equivalent formulations have been introduced including the adjoint equation [42, 53] and the path space equation [54]. The particular formulation is not important as the essence of the problem remains the same. The radiance appears both inside and outside the integral because the radiance at a point depends on the radiance at many other locations and vice-versa. This is what makes the problem inherently global and inspired the name global illumination. Also note that before we can begin, three functions must be specified as the input: the geometry G, the light scattering f_{s}, and light emission L_{emit}.

One advantage that we have is that, under our assumptions, the global illumination problem is linear in many respects. For example the problem is linear with respect to the

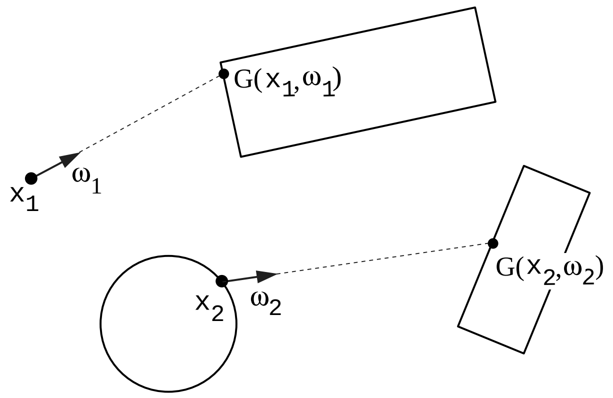

Figure 2.10: The geometry function $G(\mathbf{x}, \omega)$ returns the point which is the first intersection of the corresponding ray with a surface. It is closely related to the distance function shown in Figure 2.6.

light sources. The radiance due to two light sources is simply the sum of the radiances due to each of them individually. This linearity assumption along with the that of the BSDFs, allows us to simulate the lighting using a simple non-interacting particle system. Linearity is an extremely useful property that we want to exploit whenever possible to simplify the solution process.

2.3.3 Image Formation

Typically the results of the global illumination are displayed to a user as a set of one or more images. The pixels that make up each image are samples[9] of the tristimulance function L_{t}, hence it is these values that we will ultimately need.

This process of forming an image from the tristimulance function is illustrated in Figure 2.11. A camera model along with the constant radiance assumption, defines a mapping from parts of the tristimulance function on surfaces to pixels in our desired

[9]More generally each pixel is the integral of a filter function with the tristimulance function over a small region. For simplicity we will assume that our pixels are point samples of the tristimulance.

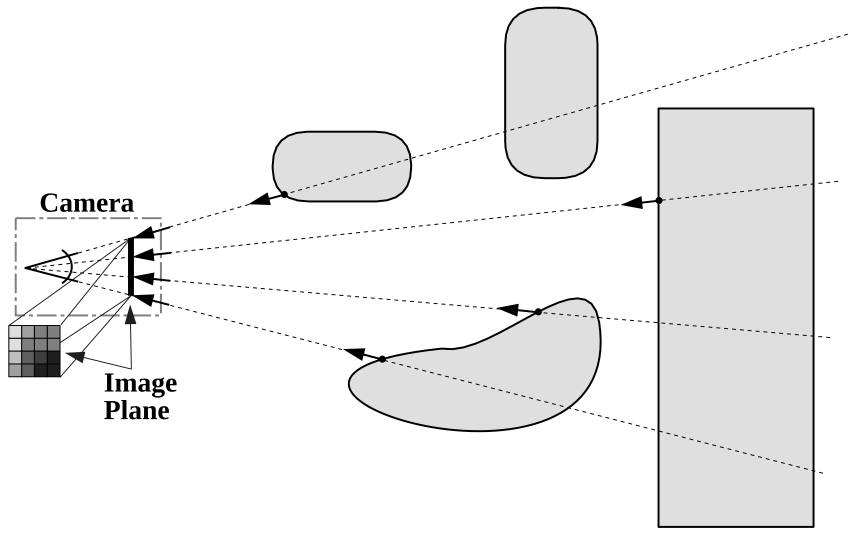

Figure 2.11: The camera defines a mapping between pixels on the image plane and values in the tristimulance. These tristimulance can further be mapped back to tristimulance values at surfaces. (Remember that tristimulance is our name for the tristimulus values of the spectral radiance.)

image. Finding these tristimulance values, though, is much more difficult. From Equation 2.13, in order to find the tristimulance, we will need to know something about the spectral radiance. From Equation 2.28 we can see that in order to obtain the spectral radiance in one place, we need in general information about the spectral radiance in many other places (Figure 2.12) and each of them depends on yet other places.

Reducing the spectral radiance to its tristimulus values (i.e. the tristimulance in our terminology) simplifies it considerably. Thus it is tempting to perform all our global illumination calculations in terms of the tristimulance. However a perceptual function such as the tristimulance only contains enough information to predict how the light will interact with a human observers' eyes. It may not accurately predict how light will interact with other objects such as the scattering at a surface and a premature conversion to perceptual measures can lead to unwanted error. Many current rendering systems fail to properly distinguish between proper uses for physical and perceptual spaces. We can legitimately neglect parts of the spectral radiance (e.g., infrared light) that we know will not affect the final tristimulus values.

2.4 Previous Work

In discussing previous global illumination algorithms and their relation to the density estimation framework, it will be helpful to introduce some classification criteria.

2.4.1 View-Independence vs. View-Dependence

One useful way to classify global illumination algorithms is by their degree of view-dependence vs. view-independence. Since a particular view or image may need only a small part of the total tristimulance function, one approach is to try to only compute as small a portion as necessary. A pure view-dependent method stores tristimulance values only on the image plane and computes the minimum information necessary to find these

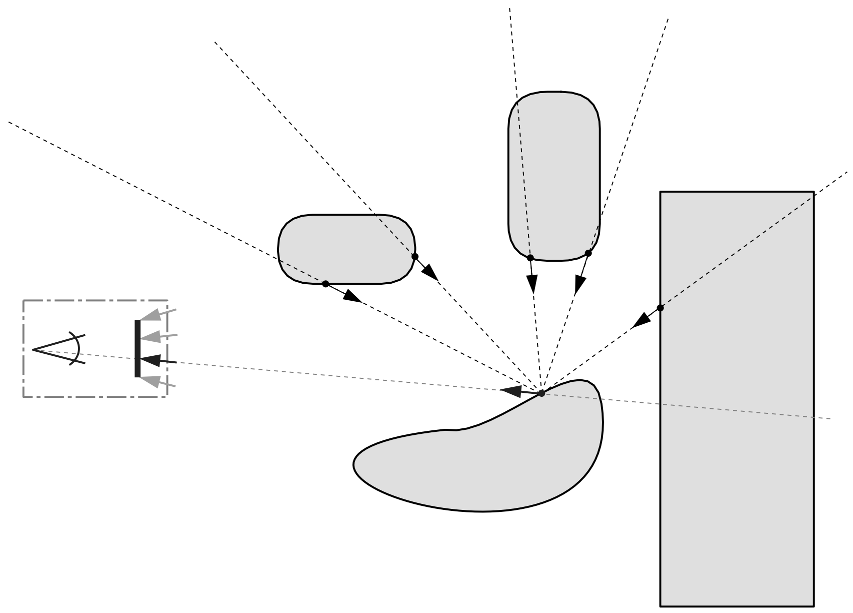

Figure 2.12: According to Equation 2.28 the radiance at a point depends on the radiance at all other points which are visible from that point. Their radiances, in turn, depend on all points visible to them and so on ad infinitum. This is what makes global illumination both global and difficult.

values. One disadvantage is that as shown in Figure 2.12, finding the tristimulance at one point generally involves computing radiance information at many points. Because these intermediate results are not stored, they may have to be recomputed many times in the process of finding all the tristimulance values needed to make up an image.

View-independent methods take the opposite approach. They do not try to compute a single view, rather they attempt to compute and store a complete global radiance function, usually the tristimulance. Once this is known, the individual tristimulance values can be quickly queried and the image from any viewpoint quickly generated[10]. If we are only interested in a small number of views, we may compute information that is not needed for any of our views. But the largest problem is that the complete radiance function is in general extremely complex. It is not clear that we can estimate it adequately or that we could store it even if we could find it. In practice these methods actually compute a simplified form of the global radiance that is limited by our need to be able to store and manipulate it conveniently.

In summary, view-dependent methods have the advantage that they require minimal storage, need to make few simplifying assumptions, and do not compute parts of the radiance which are not necessary. Their primary disadvantage is primarily in efficiency due to the potential for large amounts of redundant computation and difficulty in finding certain types of lighting features such as caustics.

View-independent methods allow all of the illumination to be precomputed which allows quick generation of arbitrary views and interactive walkthroughs of static environments. Their disadvantage is that the sheer complexity of a complete radiance function leads to an explosion of resource usage and requires simplifying assumptions to make it tractable.

Many current approaches are hybrids that attempt to combine the strengths of both view-independent and view-dependent methods. Typically they use a low resolution view-independent first pass to accelerate a view-dependent second pass. The low resolution

[10]This presumes a static model so that the radiance function does not change

greatly simplifies the view-independent computation. Using the results of the first pass, the second pass avoids much of the redundant computation and reduces the artifacts caused by the initial low resolution.

When a suitable partition of work between the passes can be found, hybrid methods are often the fastest way to compute individual images, but it is not always obvious how to partition the work. Also the view-dependent pass is still currently too slow for interactive walkthroughs, and the interaction between the passes greatly complicates the error analysis.

In this thesis we will emphasize the use of the our density estimation framework as a view-independent method. When used in this fashion, we will be able formulate a relatively precise error model and to perform interactive walkthroughs. This framework can also be used as the view-independent part of a hybrid approach and we will discuss that use briefly as well.

2.4.2 Transport vs. Representation

To understand our framework, it will be helpful to introduce a conceptual split of view-independent global illumination simulation into two distinct pieces: light transport and lighting function representation. The light transport is the flow of light between surfaces and throughout the environment. Since light from each surface potentially interacts with every other surface, we say the problem has a high inter-surface or *transport complexity*. Representation is the process of reconstructing and storing an estimate of the a radiance function on each surface. For a purely view-independent method, we want these estimates to be sufficiently detailed for direct display, so we need to be able to capture complex features such as shadows and caustics. Thus, whatever representation we choose (e.g., piecewise linear or wavelet), it must be complex enough to capture these features if they exist. Thus it has a high intra-surface or *representation complexity*. The combination of transport and representation complexity is one of problems that makes global illumination particularly challenging.

The clean separation of these two parts is a chief characteristic and advantage of the density estimation framework. The split allows us to optimize each part individually. The transport can be handled with considerable generality. Simplifications and approximations needed for quick display and compact storage can be delayed until the representation stage without compromising the accuracy of the transport phase.

2.4.3 Finite Element Methods

Currently the most widely used view-independent global illumination methods are the finite element methods[11]. Their general approach is outlined below.

If we had a guess L'_λ for the true spectral radiance function L_λ in theory we could verify its correctness by seeing if it satisfies Equation 2.28. We would evaluate the righthand side of the equation, assuming for the moment that we could compute the integral, as:

$$L''_\lambda(\mathbf{x}, \omega', \lambda) = L_{\text{emit}}(\mathbf{x}, \omega', \lambda) + \int_{\Omega_\odot} f_{\text{s}}(\mathbf{x}, \omega, \omega', \lambda) L'_\lambda(G(\mathbf{x}, \omega), -\omega, \lambda) \, |\omega \cdot \hat{\mathbf{n}}| \, d\omega \qquad (2.29)$$

If this matches the lefthand side of Equation 2.28 (i.e. $L'_\lambda = L''_\lambda$) then L'_λ is the correct radiance function. Even if L'_λ is not correct, under some mild conditions L''_λ (i.e. the function we found by substituting into the righthand side of Equation 2.28) will be closer to the true spectral radiance function than L'_λ was. By repeatedly applying this process, we can generate a sequence of ever improving radiance function estimates. This is the basic idea that underlies the current finite element approaches to the view-independent global illumination problem. This approach however, requires two assumptions which are somewhat problematic. First, we need to assume that we can adequately represent these

[11]In the graphics community these methods are often refered to as “radiosity” methods, but this name is really a misnomer. Radiosity is an older name for the radiant exitance. While some finite element methods estimate the radiance exitance, others estimate different functions such as the irradiance or tristimulance. In addition there are non-finite element methods that also estimate the radiance exitance. The name finite element methods is more descriptive.

radiance function estimates which may be quite complex. Second, we need to assume that we can adequately evaluate this complex integral.

Finite element methods work by choosing a finite set of basis functions. Only functions which are combinations of these basis functions can be represented as a set of corresponding coefficients. Of course these basis functions will only span a small subset of the space of all possible functions. Each time we generate a function, we will have to project it into our representable space adding some amount of error. A large number of basis functions are required to reduce the projection errors and allow for sufficiently detailed estimates which creates a high representation complexity.

At each iteration we will estimate the integral in Equation 2.29, or more typically some subset of this integral, project the results into our representable space, and update our current radiance function estimate. When evaluating the integral each basis function potentially interacts with every other basis function which results in a high transport complexity. Because each iteration mixes elements of both representation and transport, it suffers from their combined complexity. It also means that we may need to access the entire representation which must be kept it in local memory for good performance. This causes severe memory constraints that are often the limiting factor in such computations.

This process can be accelerated considerably by finding ways to reduce either the transport or representation complexity. One way to reduce the transport complexity is by approximating some weak interactions between groups of basis functions as a single aggregate interaction. Examples of this include hierarchical [23], wavelet[21], and clustering [50] methods. These can reduce the transport complexity, but require additional data structures that further increase the memory requirements.

Many different approximations have been used to reduce the representation complexity. The most current view-independent methods, including most finite-element methods, ignore the distinction between spectral and perceptual space and compute Equation 2.28 using the tristimulance rather than the spectral radiance. This simplifies the representation considerably, but may not be physically accurate. The error introduced is difficult to predict and depends on the spectral complexity of the emission and BRDF functions.

Another common approximation is to assume that all the BRDF and emission functions are Lambertian or diffuse. Under this assumption there is a simple relationship between the radiance and the radiant exitance at surfaces given by Equation 2.25. In this special case it is sufficient to compute and to store the simpler radiant exitance rather than the higher dimensional radiance function. Without the directional information of the complete radiance function though, a finite element method cannot correctly handle non-Lambertian materials such as metal, glass, or glossy surfaces.

An increasingly popular approximation is to abandon the idea of directly displayable results and move toward hybrid methods. The view-independent solution is computed at a very coarse resolution to keep the representation complexity down. If viewed directly this would lead to disturbing artifacts and low quality images. Instead a view-dependent pass is used to fill in much of the higher frequency information which is missing from the low resolution solution. This greatly lowers the representation complexity but has the disadvantages listed previously for hybrid methods.

In theory we could represent the radiance function with very few coefficients if we could somehow pick basis functions that were specifically tailored toward the true radiance function. Therefore another approach is to try to precompute some information about the radiance function in order to choose a better set of basis functions. The principal example of this approach is discontinuity meshing [36] which attempts to precompute the locations of shadow boundaries. This approach can dramatically improve targeted features such as primary shadows. However for general features, it is not clear that predicting such features is any easier than solving the global illumination problem in the first place. Thus this approach tends not to generalize very well to complex situations.

Another way to deal with the high combined complexity is to try to partition the environment into small, weakly interacting subsets. If such a partitioning can be found then the subsets can be solved semi-independently [52] and the computation can be ordered to use virtual memory efficiently. The difficulty is that such a partitioning is not always possible, particularly in a large open environment such as a hotel atrium.

In summary, the basic finite element method is conceptually simple and appealing, but suffers from several drawbacks. Researchers have devised a variety of techniques that greatly reduced many of these drawbacks, but they do not work in all circumstances and each one adds additional complexity to the basic method. Writing a state of the art finite element system is a major software engineering undertaking. We believe that it is still interesting and useful to explore alternate approaches toward view-independent global illumination.

2.4.4 Monte Carlo

Monte Carlo methods are widely used for attacking global illumination and other difficult problems. Finite element approaches simplify the problem of handling arbitrary functions by restricting themselves to working in the much simpler and smaller space of functions spanned by the set of basis functions. The Monte Carlo approach is quite different. When evaluating a difficult function or integral, a Monte Carlo method uses random numbers to make a guess at its value. In general this guess will be wrong, but has the very special property that it will be right on average. Thus one can get an arbitrarily good answer by averaging together enough of these guesses. Monte Carlo methods are useful because it is often much easier to be right on average than it is to be right, especially for complex and high dimensional problems.

The error in a Monte Carlo (or any other) method can be separated into noise and bias. Each time a Monte Carlo method is executed it will in principle produce a different answer and this variation in the results is called the noise. The noise can always be reduced by averaging together many results. In general the noise will fall off inversely with the square root of the number of samples (i.e. you need to average four times as many estimates to cut the noise in half).

The bias is the part of the error that would be remain even after averaging an infinite number of samples (i.e. eliminating the noise). Deterministic methods, which always produce the same result given the same input, by definition produce no noise and all of their error is bias. The results of an unbiased Monte Carlo method, such as was discussed above, contain noise but no bias. Monte Carlo methods can also contain bias if the results are not quite right on average.

Among Monte Carlo advocates there is a justifiable prejudice against biased Monte Carlo methods. Frequently it is quite difficult to estimate the nature or magnitude of such bias while noise is much easier to understand and measure. The density estimation framework is a Monte Carlo method which allows some bias in order to reduce the noise. In our case, though, we understand and can precisely formulate the nature of our bias.

2.4.5 Density Estimation Framework

In designing the density estimation framework [47, 56] we wanted to explore methods that would not suffer from the same problems as current finite element methods. One idea we wanted to explore is reducing the amount of complexity that must be dealt with at one time, by splitting the computation into distinct and separate transport and representation stages. During the transport stage we compute the flow of light within our environment via particle tracing without ever explicitly attempting to reconstruct a radiance function.

The representation stage then explicitly reconstructs an estimate of the tristimulus exitance, using information recorded during the transport stage. The lighting intensity on a surface is reflected in the density of light particles that strike the surface. Therefore our reconstruction problem is a density estimation[12] problem.

Particle tracing has been used by many other researchers [1, 2, 42] to compute light flow and Heckbert [27] first noted that reconstructing lighting from particles is a density

[12]Density estimation is the problem of estimating a unknown density function from a set of discrete samples drawn according to the density function [49]. This is a common statistical problem that occurs in many fields.

estimation problem. Since then a variety of different density estimation techniques have been applied, including histograms [27], kernel methods [10, 12], and splines [44]. Our density estimation technique is kernel based and includes several new innovations, but the most significant difference between our framework and previous work is the complete separation of the transport and reconstruction stages.

The particle tracing stage computes an unbiased statistical simulation of global light transport. By our linearity assumptions in formulating the light transport equations, the light does not interact with itself. This frees us from needing an explicit estimate for the radiance function and each of our particles can be traced independently. Instead some particle history information is recorded for later use. Thus the particle tracing can work directly with the raw input data, and deals only with the transport complexity without the representation complexity. As such it can easily handle complex geometry, BRDF, and emission functions.

The lighting reconstruction stage uses the recorded particle histories to estimate the lighting function on each surface. Because all of the global transport was handled in the previous stage, we can now reconstruct the lighting on each surface independently. The absence of transport complexity means that we can divide the reconstruction problem into pieces that have a manageable representation complexity. An additional benefit is that we can take a truly non-parametric approach in the reconstruction that delays making any decisions or assumptions about our target function until we have collected as much information as possible. This allows us to produce better reconstructions from the particle data, a definite advantage as the particle tracing is still the most expensive part of the process. We will allow some bias in the reconstruction, but because of the separation, it is purely local in effect.

By separating the transport and representation complexity into different stages, the individual stages require fewer resources than finite element methods, especially in terms of memory. But we have not completely escaped the complexity problem. The combined complexity is contained in the voluminous particle history data which is far larger than the final results will be. Thus, careful attention must be paid to how the particle histories are stored and accessed. The particle tracing only writes particle data and the density estimation only needs the particle data associated with one surface (or piece of a surface) at a time. The only operation we ever need to perform on the entire particle data set is a sort by surface; otherwise, the computation can be structured so that the particle data is processed in a largely predictable and localized order. This means that we can efficiently store the particle data using secondary storage such as a hard disk, which is typically 50 to 100 times cheaper than physical memory (RAM). The efficient use of secondary storage is one of the key ideas that makes our framework feasible. A more detailed description of the density estimation framework will be given in Chapter 3.

2.4.6 Diffuse vs. Non-Diffuse

The extent to which the density estimation framework can handle non-diffuse materials is a common point of confusion because it differs from most previous methods. The separation of transport and reconstruction phases allows us to use different assumptions in each phase. The transport phase can simulate the full spectral radiance and includes the influence of non-diffuse surfaces. In our implementation, the reconstruction phase converts to tristimulus space and introduces a Lambertian approximation but only after the transport phase has completed.

Here we will only reconstruct the tristimulus exitance even though we simulate the full spectral radiance. During the reconstruction we will approximate each surface as being Lambertian and only reconstruct the tristimulus exitance. This also means that our reconstruction will not contain complete information on non-diffuse surfaces. Displaying these surfaces correctly requires additional processing to fill in the missing directional information. Frequently we simply do not attempt any reconstruction at all on the non-diffuse surfaces. Note that unlike finite element methods, our representation approximations do not affect the accuracy of our transport. Figure 2.13 illustrates the

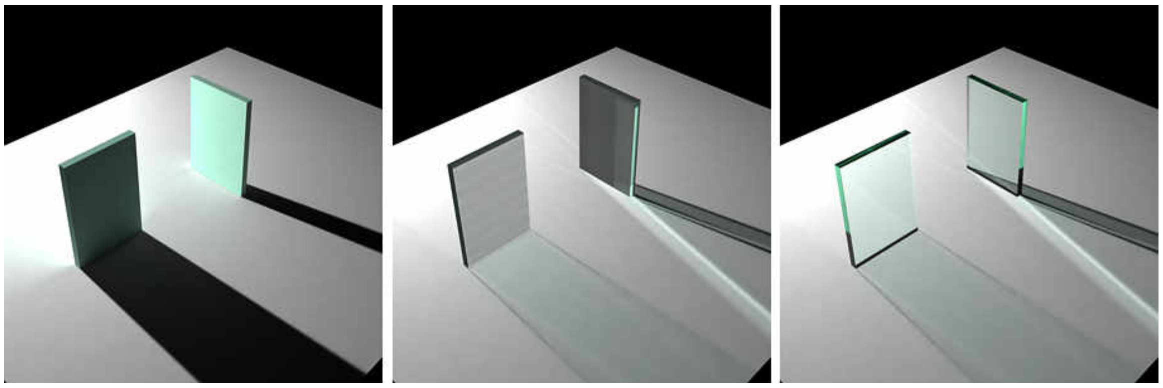

Figure 2.13: Three solutions for two glass planes on a base plane. (Left) Finite element methods typically use a Lambertian-only approximation for both transport and representation. (Center) The density estimation correctly handles general transport but then uses a Lambertian-only representation. (Right) Further adding a view-dependent pass for non-Lambertian surfaces produces the correct image.

distinction between transport and representation/reconstruction approximations.

In order to work with the simpler tristimulus exitance function, most finite element methods require that all BRDFs must be Lambertian along with a premature conversion to tristimulus space. Our framework can use such simplifications even in the presence of non-Lambertian and/or spectrally complex BSDFs without compromising the the accuracy within the transport simulation.

We could actually try to reconstruct the full tristimulance rather than the tristimulance exitance, but this would entail several difficulties. Since the tristimulance is a higher dimensional and more complex function, it would require significantly more particle data to estimate adequately. To store it compactly and display it quickly, we would need to find good ways of representing the tristimulance with its directional as well as spatial dimensions. Lastly, would need to update our reconstruction and decimation techniques to handle the tradeoff between spatial and directional resolution. Therefore this has been left as future work.

2.4.7 Hybrid or Multi-Pass Methods

A second approach toward non-diffuse surfaces is to use the density estimation framework as part of a hybrid or multi-pass technique. A ray-tracing based view-dependent pass is added to generate images. For diffuse surfaces, this pass will directly display the results computed by the density estimation process, but for non-diffuse surfaces it will trace additional rays in order to estimate their view-dependent directionally-varying appearance. The center image in Figure 2.13 shows the results of the density estimation while the right image used a view-dependent pass to fill in the appearance of the glass blocks themselves. We will use this hybrid technique for a few of the images in this thesis, but will not include such a view-dependent pass in our error analysis.

Many researchers feel that hybrid (or multi-pass) methods (e.g., [10, 45, 50, 31]) are the most promising of global illumination approaches. While we do not necessarily disagree with this assessment, we have chosen to emphasize the density estimation framework as a purely view-independent technique for several reasons.

Hybrid methods split the lighting into various components, some of which are precomputed once by the view-independent phase while others are computed anew for each image. Generally they try to divide the lighting into low and high frequency components by using distinctions such as direct vs. indirect lighting and specular vs. diffuse reflection. Each component is computed by the method which seems best suited to its expected characteristics.

The work of Jensen [31] is particular relevant here as he makes use of a particle tracing phase similiar to ours. He uses two particle tracing phases to estimate two lighting components: diffusely reflected indirect light and specularly reflected indirect light. A view-dependent phase then performs density estimation to query these components, filters the results through a local-gather operation, and combines them with estimates of the other lighting components. This method often works quite well especially when the distinctions between different lighting components are clear, and is much faster than the framework presented here when only a small number of views are needed.

Nevertheless there are also many cases where hybrid or multi-pass methods are not currently suitable. Their view-dependent passes are currently too slow for interactive walkthroughs even for static environments. We also believe that the kinds of distinctions used in multi-pass methods are sometimes difficult to make. For example, if we move away from "bare bulb" lights and direct sunlight toward complex lighting fixtures and areas lit by reflected sunlight, the distinction between direct and indirect light becomes unclear and unhelpful. Thus it is interesting to explore methods that do not rely on such distinctions. Lastly the heterogeneous nature of hybrid methods makes their error analysis extremely difficult. By computing all of the transport using a single and simple statistical particle tracing approach, we can more easily and precisely analyze the error in our solutions. We hope that availability of a precise error analysis will enable some new applications of global illumination and can also be used as feedback to improve the solution process itself.

A few other difference between our work and Jensen's are worth mentioning. Jensen does not ever explicitly reconstruct the illumination function on a surface in our sense. Instead a new density estimation calculation is performed each time he wants to query information computed by particle tracing. To make this feasible, he needs to be able to keep all his particle data in memory, which puts a stringent limit on the number of particles. He also uses a very simplistic density estimation technique, which suffers from errors due to boundary effects, surface curvature, and nearby surfaces, but the resultant artifacts are mostly filtered out by his local-gather display computations. In our work we need to be more careful since our density estimation results are directly displayed.

Chapter 3

The Density Estimation Framework

3.1 Overview of Framework

The density estimation framework is divided into three principal phases: particle tracing, density estimation, and decimation. As discussed in Section 2.4.2, we separate the view-independent global illumination problem into light transport and light representation. By transport, we mean simulating the total flow of light throughout the target environment. Representation, on the other hand, is the process of finding and storing usable estimates of lighting or radiance functions on individual surfaces. We first simulate the complete transport in the particle tracing phase before we attempt to reconstruct appropiate representations.

To simplify our task, we further split the representation phase into two pieces: density estimation and decimation. The goal of the density estimation phase is to reconstruct the best approximation possible. The decimation phase then optimizes these results for compactness while maintaining their perceptual quality. In between the transport and representation stages, a sorting step is used to reorder the particle data for efficient access during density estimation.

The three phases can be summarized as follows:

1. **Particle-tracing phase:** Power-carrying particles, each with a specific wavelength, are emitted from the luminaires according to their emitted spectral radiance distribution, L_{emit}. They are tracked as they travel through the environment until they are absorbed. Each time a particle hits a surface it is probablistically absorbed or scattered in a new direction according to the BSDF f_{s}, or Bidirectional Scattering Distribution Function, of the surface. A list of "hit points" where particles hit surfaces is generated and saved.

2. **Density-estimation phase:** The stored hit points are used to reconstruct approximate lighting functions on each surface. The intensity of illumination is related to the density of the hit points so this is a density estimation problem. To reconstruct the tristimulus exitance, the hits are weighted by the surface reflectance[1] and the CIE XYZ response functions and then their density is estimated at a set of points using local linear density estimation. The result is a Gouraud-shaded, or piecewise linear, mesh of triangles with a color at each vertex suitable for direct display.

3. **Mesh decimation phase:** The initial mesh is generated conservatively and in most places is more dense than required. The decimation creates a sparser mesh while minimizing the changes in its appearance. The mesh is decimated by progressively removing vertices as long as the resulting change is below a perceptually-based threshold. The simplified mesh is more compact and faster for display.

3.1.1 Some Benefits

There are a number of advantages to dividing the view-independent global illumination problem into these three phases. Each phase is less complex, both conceptually and computationally, than the original problem. The high complexity of global illumination makes divide-and-conquer strategies quite appealing when the divisions can be made

[1]Or more properly by the surface's BSDF on non-Lambertian surfaces.

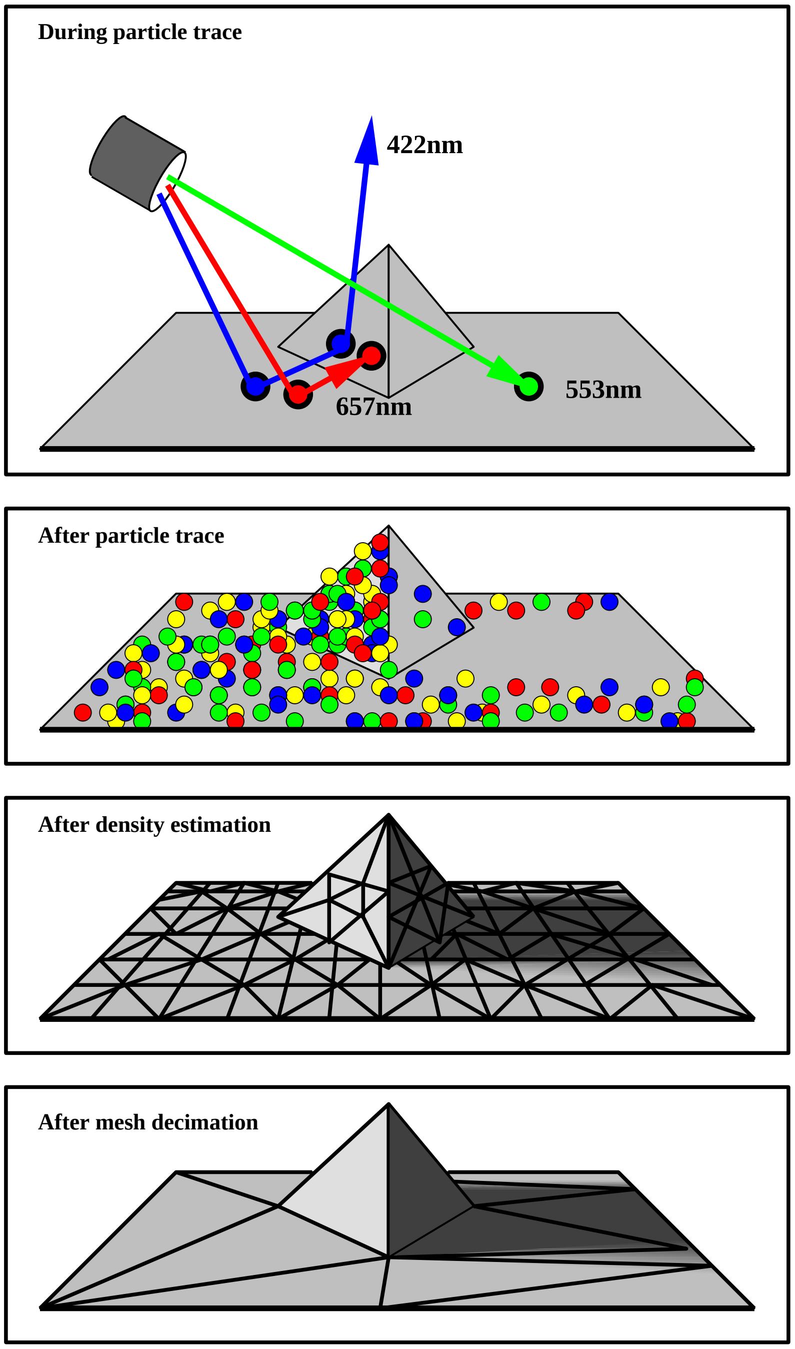

Figure 3.1: Overview of the density estimation algorithm showing the type of results produced by each phase.

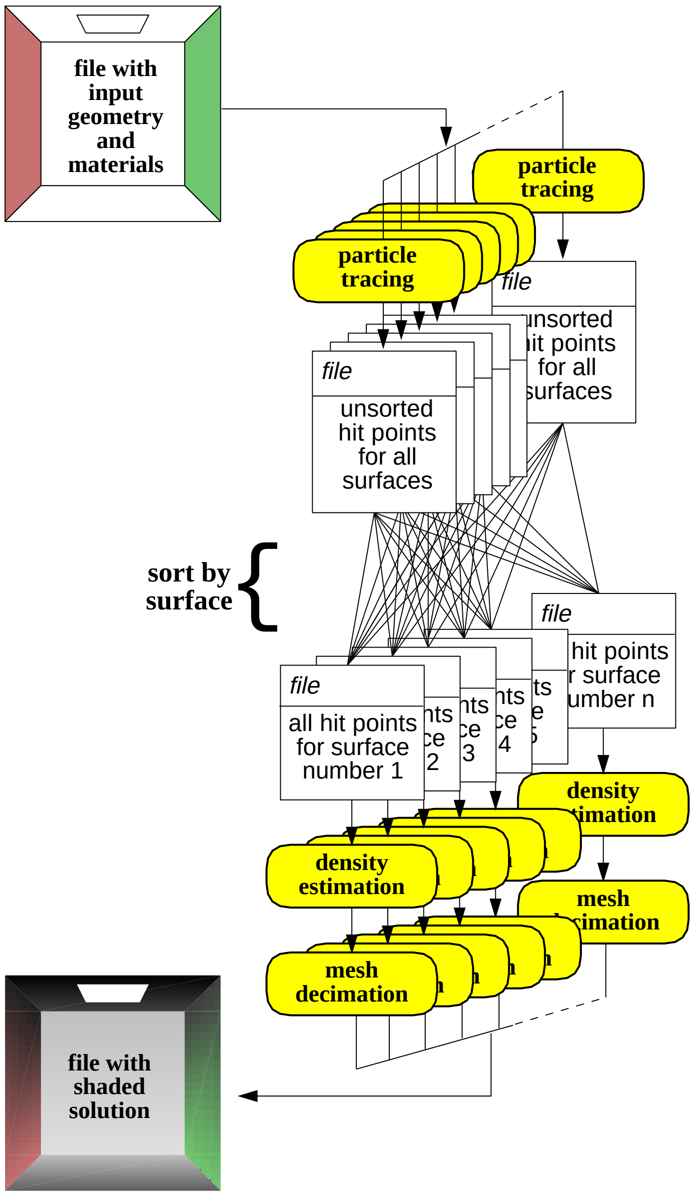
file with input geometry and materials
particle tracing
particle tracing
file
unsorted hit points for all surfaces
file
unsorted hit points for all surfaces
sort by surface
file
all hit points for surface number 1
file
hit points for surface number n
density estimation
density estimation
mesh decimation
mesh decimation
file with shaded solution

Figure 3.2: Overview of the density estimation algorithm showing the flow of data through the phases and the inherent parallelism.

cleanly as is done here. As we argued in Section 2.4, the separation of transport and representation complexity reduces the overall computational complexity.

The division also allows us to use different assumptions and approximations in each phase. They can be optimized for their particular sub-goals in pursuit of the total goal without altering the other phases. For example we can use a simplifying approximation, such as all surfaces are Lambertian, in the density estimation phase without compromising the accuracy within the transport phase (see Figure 2.13). The particle tracer can be optimized for physical accuracy, the density estimation for perceptual accuracy, and the decimation for compactness and display speed.

The software naturally divides into three modular pieces corresponding to the three phases. Once the format for data flowing between the phases is chosen, they can be implemented, debugged, maintained, and even updated independently. In our case each phase was implemented by a different person with very little coordination required[2].

Another benefit is the inherent parallelism in this framework. Finite element methods tend to be difficult to parallelize because, they mix transport and representation operations. Our particles do not interact with each other, thus each particle could potentially be traced in parallel. Surfaces do not interact during the representation phases, thus the lighting reconstruction on each surface or piece of a surface could be performed in parallel. As shown in Figure 3.2 significant communication is only required during the sorting operation, and parallel sorting is well studied problem. Some of this potential was demonstrated in [62], and this parallelism has remained one of our design goals throughout our algorithmic improvements.

[2]The three phases were implemented by Peter Shirley, Bruce Walter, and Philip Hubbard respectively.

3.2 Particle Tracing

The goal of the particle tracing phase is to simulate the complete flow of light throughout a target environment. One of particle tracing's advantages is that it can simulate a fairly general geometric optics based model of light propagation. The two major limitations are that light is assumed to travel in straight lines between scattering events and that the scattering is a linear process. Wave effects such as diffraction and interference are not included except where such effects can be hidden inside the scattering function f_s. Our current implementation handles effects such as refraction and dispersion, and several others such as polarization and fluorescence could be added relatively easily.

Tracing of light particles is sometimes referred to as "photon tracing". While it has intuitive appeal, this name can be a little misleading. Unlike photons, our particles generally do not obey wave optics and carry an amount of power based the human eye's sensitivity to their wavelength. A one-for-one photon simulation would infeasible anyway, even if it were desirable, due to the incredibly large number of photons in real environments. The properties of our particles are often chosen for computational efficiency rather than to match physical photons.

The particle tracing consists of three basic operations: generation, ray casting, and scattering. A particle is first generated randomly at a light source. Next a ray is cast to find the first obstruction in its straight line of travel. Here it is probabilistically either absorbed or scattered in some new direction. If scattered, another ray is cast to find the next scattering event and the process is repeated until the particle is eventually absorbed. More particles are generated and traced in the same way until some stopping criteria is reached. At each step we want our particles, on average, to mimic the behavior of light so that our simulation will be faithful to reality.

The particle tracing is a Monte Carlo process and the particles are generated according a probability density function $p_e(\mathbf{x}, \omega, \lambda)$ where $\mathbf{x}$ is the initial position of the particle, ω is its initial direction, and λ is its wavelength. The power distribution of these particles must match the emitted radiance L_{emit} as specified for this particular environment. We can ensure this by setting the initial amount of power ϕ carried by the particle to be:

$$\phi = \frac{L_{\text{emit}}(\mathbf{x}, \omega, \lambda)\,[\omega \cdot \hat{\mathbf{n}}]}{N_{\text{p}}\, p_{\text{e}}(\mathbf{x}, \omega, \lambda)} \tag{3.1}$$

where $\hat{\mathbf{n}}$ is the surface normal at $\mathbf{x}$ and N_{p} is the total number of particles traced.

As before, we assume a non-participating medium so that scattering only occurs at surfaces. We find the location, $\mathbf{x}'$, of the next scattering event by casting a ray and finding the first surface it intersects. This is equivalent to evaluating the geometry function, $\mathbf{x}' = G(\mathbf{x}, \omega)$ (Section 2.3.2), and depends solely on the geometry of the target environment.

During a scattering event at location $\mathbf{y}$, a particle traveling in direction ω with wavelength λ will be scattered in direction ω' with some probability $p_{\text{s}}(\omega')$. Alternatively the particle may be absorbed with some probability which we will write as $p_{\text{s}}(0)$. Note that we have added zero as a special direction to signify the case when the particle does not leave in any direction. Again we must ensure that the power distribution of scattered particles matches that specified by the BSDF f_{s}. We can achieve this by setting the new amount of power ϕ' carried by the scattered particle to be:

$$\phi' = \phi \frac{f_{\text{s}}(\mathbf{y}, -\omega, \omega', \lambda)[\omega' \cdot \hat{\mathbf{n}}]}{p_{\text{s}}(\omega')} \left(\frac{n_1}{n_2}\right)^2 \tag{3.2}$$

where ϕ is the power carried by the particle before it was scattered, $\hat{\mathbf{n}}$ is the suface normal at $\mathbf{y}$ and n_1 and n_2 are the indices of refraction of the medium that the particle is coming from and scattered to respectively. The last factor is only needed when the particle crosses a surface into a region with a different index of refraction (e.g., goes from air to glass) and is further discussed in [53].

3.2.1 Choosing the Particle Power

Our choice of emission and scattering probabilities determines the amount of power carried by each particle and has a large effect on computational efficiency. The number of particles traveling through a region, and hence the amount of information we have about the lighting there, depends partially on the amount of power carried by those particles. For a fixed number of particles, varying the particle power allows us to trade off increased resolution of the radiance function in some regions for a corresponding decrease in other areas.

Such a trade off only makes sense when we know that certain parts of the radiance function are more important to us than others. In the absence of such prior knowledge about the radiance function, the most rational decision is to have all particles carry the same power. We can achieve this by making the emission probability, p_{e}, proportional to the emitted radiance:

$$p_{\mathrm{e}}(\mathbf{x}, \omega, \lambda) \propto L_{\mathrm{emit}}(\mathbf{x}, \omega, \lambda)\,[\omega \cdot \hat{\mathbf{n}}] \tag{3.3}$$

Since p_{e} is a probability density function whose total volume must be equal to one, this completely determines p_{e}. Similiarly to maintain constant particle power, the scattering probability p_{s} for an particle being scattered at location $\mathbf{y}$ that was previously traveling in direction ω with wavelength λ will be:

$$p_{\mathrm{s}}(\omega') = f_{\mathrm{s}}(\mathbf{y}, -\omega, \omega', \lambda)[\omega' \cdot \hat{\mathbf{n}}]\left(\frac{n_1}{n_2}\right)^2 \quad \text{if } \omega' \neq 0 \tag{3.4}$$

The requirement that p_{s} be a probability density function[3] then determines $p_{\mathrm{s}}(0)$, the probability of absorbtion.

When we do have some knowledge about the relative importance of parts of the radiance function, we can use that to choose different probability functions. Some strategies for allowing the particle power to vary will be discussed in Section 6.1.

[3]Strictly speaking p_{e}, as we have defined it, is not a probability density function because we used its value at a single point, zero, to denote a non-infinitesmal probability (i.e. the probability of absorbtion). One can define p_{s} more rigorously using the formalisms of measure theory [7] or the Dirac delta function notation [37] from the physics literature.

3.2.2 Recording Particle Hits

The purpose of the particle tracing is to generate data for use by the density estimation phase. As we are interested in the illumination at surfaces, we record the interactions between particles and surfaces.

We could record either all incident particles to a surface or only the particles which are scattered and leave the surface. The scattered particles more closely correspond to our target function, the tristimulus exitance, but the incoming particles are a better choice. They form a superset[4] of the outgoing particles They correspond to the incident radiance, but we can weight them by the BSDF f_s to estimate the the reflected light. It also allows us to reconstruct the irradiance or intristimulance when we wish (e.g., at textured surfaces).

Each time a particle strikes a surface we record a surface identification number (id), the location of the hit, and the wavelength of the particle. The surface id is encoded using four bytes which allows for up to 4 billion surfaces. The x, y, and z coordinates of the hit are recorded using two bytes each using a fixed point represenation relative to the bounding box of the surface. This is sufficient to give sub-millimeter accuracy for surfaces smaller than 60 meters in length. Large surfaces could always be subdivided if more accuracy is necessary. The wavelength is stored as a two byte integer measured in nanometers. In total the particle tracer writes 12 bytes to disk for each particle hit. Each hit point file also contains the total emitted power and the number of particles used to generate it. This allows us to compute the power carried by each particle even when combining the results from multiple executions of the particle tracer.

[4]Except for newly emitted particles of course, but we assume the emission function L_{emit} is already known.

3.3 Sorting

The density estimation needs to be able to rapidly identify all hit points on a surface that are near an arbitrary point. This is a problem as the hit points, as produced by the particle tracer, are quite voluminous and in a nearly random order. To make these operations quick, we first introduce a sorting step to reorder the particle hit data for efficient localized access and to minimize the number times data must be loaded into memory from disk.

Previously [47, 56] we only required that the particles be sorted by surface id. We have since changed the sorting to enforce a more structured ordering. The newer format is necessary for our improved adaptive density estimation techniques and also makes load balancing easier in parallel implementations.

The sorted data is organized in a three level hierarchy (Figure 3.3). The data is first divided into files which are small enough to be completely loaded into memory for processing. The files are then further divided into the loadable segments that will be individually loaded and cached in memory as needed during the density estimation phase. For effective caching, the loadable segements must be small enough that many of them can fit in memory at once. Finally the data in each loadable segment are organized using a regular grid for efficient access to local regions of the data.

We expect that the total particle data will be larger than the available memory (RAM), so our first task is to partition the data into files that can fit in memory. We accomplish this using a statistical partitioning based on a random subset of the data points. This data is then partitioned into files using a single pass to through the particle data. The actual size of these files will vary somewhat from what we originally estimated, but these variations should be small. For example, if we have 300 million data points and we use 3 million of them to create a partitioning scheme such that each partition should have around 3 million data points, the relative standard deviation in the actual partitioned file size will be less than 1%, which is easily accommodated.

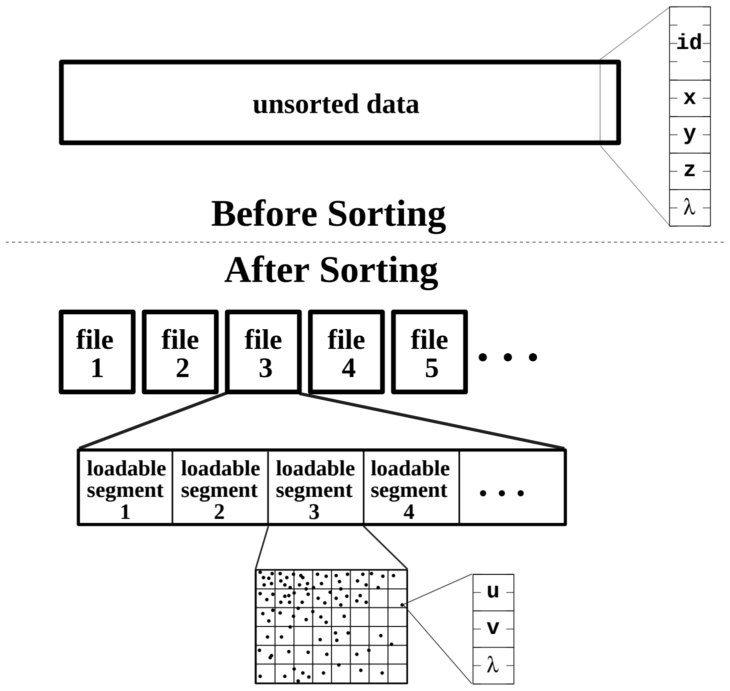

Figure 3.3: Before sorting the hit points are in a random order and encoded using twelve bytes per point. After sorting the data is organized into a three level hierarchy of files, loadable segments, and regular grids and encoded using only six bytes per point.

Our partition scheme can combine points from multiple surfaces into a single file and split the hits from a single surface across multiple files as needed to meet the target file size. Whenever a surface needs to be split, we compute a local 2D coordinate system, and split by whichever coordinate has a greater extent. The median coordinate value is used to split the surface into pieces with equal numbers of hit points. Note that this splitting is for storage and organization purposes only. The density estimation process still uses all hits associated with a surface regardless of such splitting.

After the initial partition is complete, each file is loaded in turn and divided into loadable segments based on a suitable smaller maximum size. Each loadable segment contains data from only one surface and may require further splitting of surfaces. Finally the points within each loadable segment are organized using a regular grid in the local 2D space for rapid access to the points in any vicinity.

During the sorting process we also convert the hit points to a more space efficient form. We can drop the surface id because hit points are grouped by surface. Also we translate from 3D space to a local 2D space. The resulting u and v coordinates are stored as two bytes each in a fixed point format relative to the bounding box of the surface piece corresponding to its loadable segment. Together with the wavelength, each point is encoded using six bytes after sorting instead of the original 12 bytes. This compact and highly organized format allows to rapidly and efficiently access the particle data as needed during the density estimation phase.

3.4 Density Estimation

The goal of the density estimation phase is to estimate the tristimulus exitance function as best we can from the information recorded during the particle tracing phase. If the particle tracing is performed correctly, then there is a simple relation between the spectral radiance and the density of the recorded particle hits. Let $p(\mathbf{x}, \omega, \lambda)$ be the probability that a particular hit occurred at location $\mathbf{x}$ due to a particle traveling in direction ω with wavelength λ. If each particle carries the same amount of power ϕ, then the probability density p of the particle hits is:

$$p(\mathbf{x}, \omega, \lambda) = \frac{|\omega \cdot \hat{\mathbf{n}}|}{\phi n} L_{\mathrm{i}}(\mathbf{x}, -\omega, \lambda) \tag{3.5}$$

where n is the total number of particle hit points recorded and the dot product is due to the projected area in the definition of radiance. The relation is to the incident radiance L_{i} because we chose to log incident particles. If we can estimate the probability density function p, we can use this relation to estimate the radiance or any related function including the tristimulus exitance that we want.

The problem of estimating a density function from a set of discrete samples distributed according to it is known as *density estimation*. Chapters 4 and 5 will be devoted to density estimation techniques, so we will defer a more detailed discussion until then.

We use density estimation to estimate the tristimulus exitance at a set of discrete points. These points are connected together into a mesh of triangles with piecewise linear interpolation over each triangle to create a continuous estimate of the tristimulus exitance. The spacing between these points is designed to be smaller than the smallest feature that the density estimation is capable of reconstructing. Where such a feature is not actually present, the generated mesh will be more dense than is actually required. Piecewise linear interpolation over a triangle is identical to Gouraud shading. This is convenient as Gouraud shaded triangles are the most widely supported primitive for fast shaded 3D graphics displays. For fastest display, however, we would prefer a mesh with as few vertices and triangles as possible.

3.5 Decimation

The decimation phase takes the meshes produced by the density estimation phase and optimizes them for compactness and quick display. The idea is replace an overly dense mesh with a sparser mesh whose appearance closely matches the original. We will only very briefly discuss the decimation here, as it is the density estimation that is the main focus of this thesis. See [47, 56] for a more complete discussion of our current decimation techniques.

Our approach is to make a sparser mesh by identifying and removing redundant vertices. First we need a metric to tell us how much we can change the tristimulus function before the change is likely to be either noticeable or disturbing. Each vertex is then evaluated to see how much change would be caused by its removal. Using a priority queue, vertices are removed in order of minimal incremental change in the visibility metric as long as each removal causes a change that is below some threshold. The result is a triangular mesh whose vertices are a subset of the original vertices, and which in most cases is much sparser.

The decimation parameters that control the threshold and visibility metric were chosen using a simple user study on a computer monitor under particular viewing conditions. Ideally the decimation would be recalibrated for each display device and viewing conditions, but in practice our initially determined values seem to work well in most cases.

The metric used in the decimation is currently based solely on luminance changes and does not include effects such as Mach bands and color shifts. It also does not use spatial frequency information because our solutions are view-independent. We do not know from what direction or distance a surface will be viewed. Better perceptual metrics would almost certainly lead to better performance and higher decimation ratios. Even so, the current decimation typically reduces meshes to approximately 15% of their original triangle count.

3.6 Improvements over Prior Implementation

The density estimation framework is part of an ongoing research project and earlier versions have been described in [47] and [56]. The basic three-phase framework has remained similiar to the one we proposed in [47], but the individual phases have been considerably improved. In [56] we introduced some improvements including:

- The particle tracing was changed from an RGB color space to spectral radiometry for better physical accuracy.
- Used a density estimation technique based on locally-weighted linear least-squares regression to handle arbitrary polygons and boundary bias problems.
- Perceptually-based heuristics and user studies provided a systematic way to choose decimation parameters.

In addition this thesis presents several further improvements including:

- Spatially varying adaptive bandwidths over surfaces for improved density estimation.
- Wavelength importance sampling in the particle tracing to increase the information content per particle hit.
- Use of a perceptual noise metrics to better control our visual error.
- Bias detection techniques to automatically detect and sharpen important features such as shadow boundaries.

Chapter 4

Density Estimation

Density estimation is a common problem that occurs in many different contexts and fields. One might be trying to estimate the geographic distribution of pet owners from a random phone survey, the automobile fatality rate by time of day from a set of accident reports, or the illumination on a surface from a light particle simulation. The essence of the problem is the same in each of these cases.

We are interested in estimating an unknown function f. What we are given is a set of data points that are distributed according to this function. In other words the probability that a data point will be found in the vicinity of $\mathbf{x}$ is proportional to $f(\mathbf{x})$ so that there is a strong connection between the local density of data points and the value of f.

This is a common problem in many contexts, and there is an entire field in statistics that is devoted to density estimation techniques [49, 57]. We can broadly classify the techniques into parametric and nonparametric approaches. Parametric approaches assume that the density function is known to be a member of some particular family of functions. We might, for example, know that the unknown function is a gaussian or a third-order polynomial. Such prior knowledge greatly simplifies the problem. We need only estimate the parameters that define the function within its family.

Often, however, we know very little *a priori* about the density function. In this case

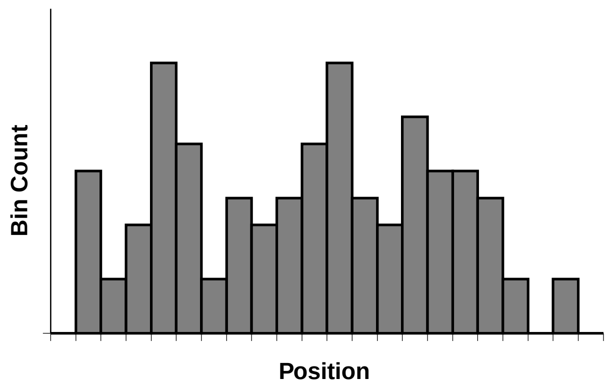

Figure 4.1: A histogram in one dimension.

nonparametric approaches are more appropriate. These try to make as few assumptions about the unknown function as possible, though typically they will prefer smoother function estimates as long as they are consistent with the input data. In our case, we expect our illumination functions to be mostly smooth, but they can contain discontinuities of all orders and need not belong to any parametric family. For us, it is best to take a nonparametric approach and let the data drive the estimation as much as possible.

4.1 Kernel Density Estimation

The histogram is probably the oldest and most widely known density estimation technique. It consists simply of dividing the domain of the problem into regions, or bins. Then we form a piecewise constant approximation by counting the number of data points in each bin and dividing by the size of the bin. The simplicity of histograms is appealing, but there are other methods that generally produce better approximations to the real density function.

In the statistics literature, kernel density estimation is one of the most widely used and recommended nonparametric density estimation technique. The density function at a point is estimated by taking a weighted sum of nearby points and dividing by the area under the weighting, or *kernel*, function. Usually the kernel is normalized to have unit area so that the division is unnecessary.

It is traditional to split the choice of a kernel function into the general shape of the kernel in a canonical form, written as K, and a kernel width parameter h, which is usually called the *bandwidth*. The kernel scaled by the bandwidth h is written as K_h and defined as:

$$\mathrm{K}_h(\mathbf{x}) = \frac{1}{h^d}\,\mathrm{K}\left(\frac{\mathbf{x}}{h}\right) \tag{4.1}$$

where d is the dimension[1] of domain. It is also traditional to assume the density function has unit volume (i.e. is a probability density function), though this can easily be generalized to other densities by adding a scaling factor[2]. Given n data locations $\{\mathbf{X}_1 \ldots \mathbf{X}_n\}$ the estimated function $\tilde{f}$ is:

$$\tilde{f}(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} \mathrm{K}_h(\mathbf{x} - \mathbf{X}_i) \tag{4.2}$$

We can think of this expression as a kernel centered at the estimation point used to weight nearby data points, or alternatively we can think of it as the sum of n kernels centered on the data points[3] and summed as shown in Figure 4.2. Thus our function estimate is simply a sum of suitably translated kernel functions. Essentially we are blurring the data points to produce a smooth estimate where the bandwidth parameter controls the amount of blurring.

[1]For our application, the domain is a two dimensional surface or more specifically a polygon, however some of our examples will be one dimensional for simplicity.

[2]This scaling factor is independent of the spatial distribution of data points and must be known from some additional information. In our case we can find the scaling factor because we know the amount of spectral power carried by each particle. For now we will assume the function has unit volume.

[3]We will break this symmetry when we introduce the local polynomial methods, but the intuition built here is still useful.

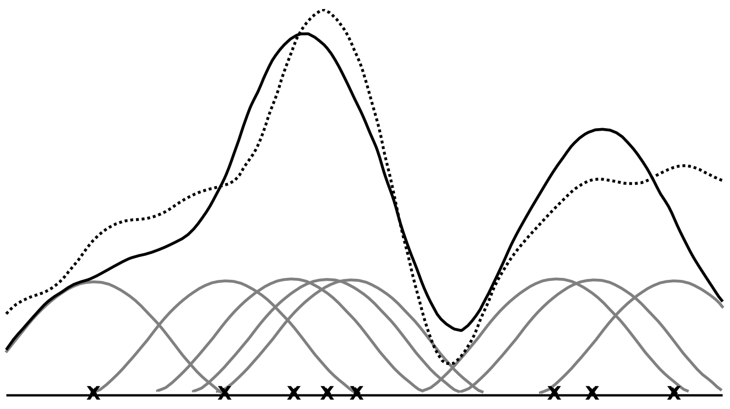

Figure 4.2: Kernel Density Estimation: data points (x's) are distributed according to a density function (dashed) and kernels (gray) centered on the data points are summed to form an estimate (solid) of the density function.

4.1.1 Choosing the Kernel

The conventional wisdom in the statistics literature (e.g., [49, p. 43]) which is borne out by our own experience [55], is that the exact shape of the kernel makes little difference in the quality of the density estimation[4]. However, we can reduce the computational costs by choosing a kernel that has compact support and is simple to calculate. The two kernels that we use are the uniform kernel and the Epanechnikov kernel.

Of all compactly supported kernels, the uniform kernel is the easiest to compute and gives the minimum variance for a given bandwidth. The equation for the uniform kernel in two dimensions is:

$$\mathrm{K}_{\mathrm{uniform}}(\mathbf{x}) = \begin{cases} \frac{1}{\pi} & \text{if } \|\mathbf{x}\| \leq 1 \\ 0 & \text{otherwise} \end{cases} \tag{4.3}$$

The main drawback of the uniform kernel is that it contains jump discontinuities, and hence so will an estimate made using it. This is not a problem if the discontinuities are

[4]In a truly nonparametric approach where we are making few prior assumptions about the function, each kernel will necessarily cover a large number of neighboring points. In this case the kernel shape is much less important than the kernel width.

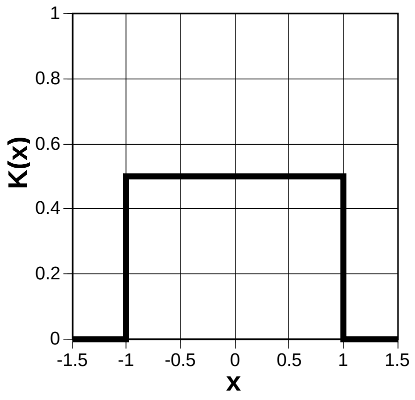

Figure 4.3: Uniform kernel in one dimension.

small enough, but we may prefer to use a smoother kernel. The Epanechnikov kernel is simple to compute, continuous, and, under some conditions, has been shown to be optimal [57, p. 30], though its efficiency advantage over other kernels is slight. The 2-D Epanechnikov kernel is given by:

$$\mathrm{K}_{\mathrm{Epan}}(\mathbf{x}) = \begin{cases} \frac{2}{\pi}(1 - \|\mathbf{x}\|^2) & \text{if } \|\mathbf{x}\| \leq 1 \\ 0 & \text{otherwise} \end{cases} \tag{4.4}$$

Many other kernels shapes are possible and have been used. If continuity of derivatives in the estimate is important one can use smoother kernels with continuous derivatives. In our application, there is nothing to be gained by using these smoother kernels.

4.1.2 Choosing the Bandwidth

Choosing a good bandwidth is much more difficult. To do this intelligently we need to understand the trade off between bias and variance. It is well known that the expected value of kernel density estimation via Equation 4.2 is [57]:

$$\mathrm{E}\,\tilde{f}(\mathbf{x}) = \int \mathrm{K}_h(\mathbf{x} - \mathbf{y}) f(\mathbf{y})\, d\mathbf{y} \tag{4.5}$$

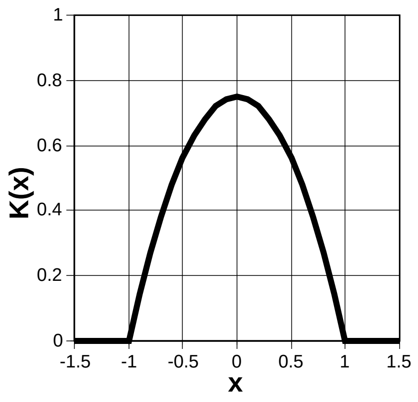

Figure 4.4: Epanechnikov kernel in one dimension.

In other words, our estimate $\tilde{f}$ converges, not to the correct function f, but rather to the correct function convolved with the kernel. The difference is bias and is one of the sources of error that we want to minimize. In general, the way to reduce the bias is to reduce the kernel size which implies we want to use as small a bandwidth as possible.

Since we use a finite amount of data, our solution will also have variance, or noise. The variance is given by:

$$\operatorname{Var}\tilde{f}(\mathbf{x}) = \frac{1}{n}\left(\int \mathrm{K}_h(\mathbf{x}-\mathbf{y})^2 f(\mathbf{y})d\mathbf{y}\right) - \frac{1}{n}\left(\int \mathrm{K}_h(\mathbf{x}-\mathbf{y}) f(\mathbf{y})d\mathbf{y}\right)^2 \tag{4.6}$$

We can get the leading term in the variance by neglecting the second term above, using the Taylor expansion, $f(\mathbf{y}) \approx f(\mathbf{x}) + (\mathbf{y}-\mathbf{x})f'(\mathbf{x})$, Equation 4.1, and assuming a symmetric kernel so that $\mathrm{K}(\mathbf{x}) = \mathrm{K}(-\mathbf{x})$ to find:

$$\operatorname{Var}\tilde{f}(\mathbf{x}) \approx \frac{\int \mathrm{K}(\mathbf{y})^2 d\mathbf{y}}{nh^2} f(\mathbf{x}) \tag{4.7}$$

From this we can see that the variance is inversely related to the bandwidth and to minimize the noise we want to use a large kernel. In essence, varying the bandwidth controls a tradeoff between bias and variance. The optimal value will depend on the local complexity of the density function f and on the relative importance that we attach to bias and noise errors. We will return to consider this bandwidth tradeoff issue more closely in Chapter 5.

Once a kernel function and bandwidth are chosen, it would be quite straightforward to implement Equation 4.2. Unfortunately, there is a fundamental problem with kernel density estimation called boundary bias.

4.2 Boundary Bias

The cause of boundary bias is that a naive kernel method does not differentiate between regions that have no data points because the density function is near zero, and regions which have no data points because they lie outside the domain where we have information about the function. Effectively it assumes that the function goes to zero everywhere outside of the domain; consequently, there is a strong bias toward zero in regions that are within a bandwidth of the boundary of the domain. In our application this would show up as a noticable darkening near the edges of surfaces or polygons (see Figure 4.5). This might not seem like a big problem because it only affects boundary regions, but in applications like ours, the boundary region can be a large fraction of the total domain. This is especially true for complex scenes, which often consist of many small polygons.

We distinguish between two degrees of boundary bias. As the bandwidth shrinks, we want the bias to converge to zero. If bias near the boundary does converge to zero, but at a slower rate than in the interior, we call this asymptotic boundary bias. If the bias does not converge to zero, we call this absolute boundary bias. The naive kernel method of Equation 4.2 suffers from absolute boundary bias. For example points on a boundary edge will typically converge to half their proper value even in the limit of many points and small bandwidths. This is unacceptable in many cases. We need methods that reduce the boundary bias problem to the asymptotic level or eliminate it entirely.

A variety of techniques have been used to reduce boundary bias. One simple technique is to create "phantom" data outside the domain by reflecting data across the boundary [48]. Unfortunately, this only partially corrects for the boundary bias and is only easy to implement for shapes which can tile the plane (e.g., rectangles). In the statistics literature, when the boundary bias problem is dealt with at all it is usually by using modified kernels in boundary regions called boundary kernels (e.g., [57, p. 47]). The difficulty with boundary kernels is that they are derived in an *ad hoc* manner, and there is no general agreement as to which are the best.

We have developed a new method for reducing or eliminating boundary bias by adapting the locally-weighted polynomial least-squares regression method described below. The results are demonstrated in Figure 4.5. The advantages of this new method are its clean conceptual basis, and that it helps to build our intuition about the behavior of kernel density estimation.

4.2.1 Local-Weighted Polynomial Least-Squares Regression

Regression is the problem of reconstructing a function given a set of function estimates at various points. It is closely related to density estimation and also suffers from boundary bias problems. In the regression literature, the locally-weighted polynomial least-squares regression technique has recently become popular, in part because it eliminates boundary bias in a natural way [25]. At each point where we want to estimate the function we fit a polynomial to the function estimates using a weighted least-squares fitting. A kernel function is used to give large weight to nearby data and little or no weight to more distant data. The value of the fitted polynomial at the estimation point is used as our estimate of the function at that point. Note that we fit a different polynomial at every point where we estimate the function.

In applying this method we are free to choose the degree of polynomial to fit. Any polynomial order will eliminate the absolute boundary bias and it turns out that odd-order polynomials do not suffer from asymptotic boundary bias. If the function is relatively smooth locally then higher order fits tend to do better. On the other hand if the

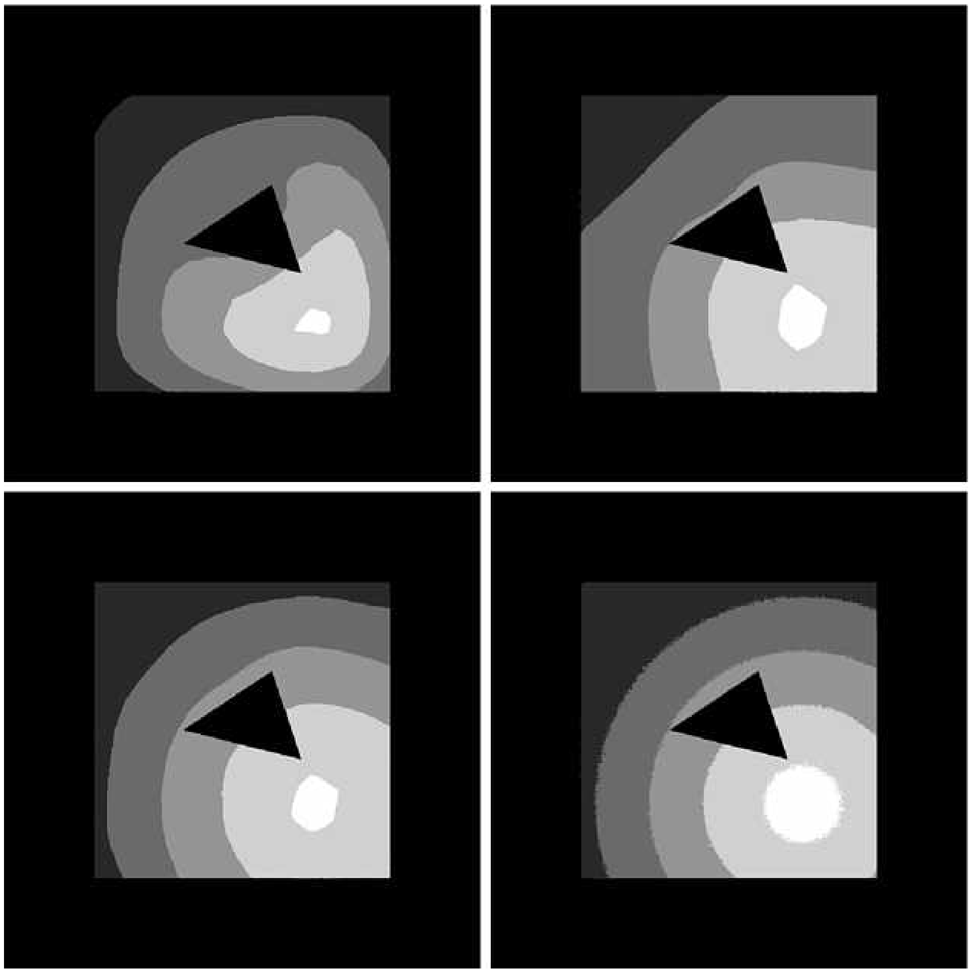

Figure 4.5: Illumination contours on a polygon with a hole. Reconstructed from $\approx$ 23,000 hit points using kernel density estimation (top left), local constant (top right), and local linear density estimation (bottom left), along with a path traced reference image (bottom right). The constant and linear methods are examples of local polynomial methods introduced in this paper.

density function changes abruptly over distances comparable to a bandwidth, then lower order fits tend to perform better. This is because higher order fits tend to require larger bandwidths to achieve an equivalent variance, while small bandwidths are the best way to capture abrupt function changes. For us, relatively sharp features such as shadow boundaries are important. Therefore we use either constant or linear fits which are zero and first order polynomials respectively. Linear fits have the advantage of eliminating the asymptotic boundary bias, but constant fits are simpler and may be preferable in some cases. We will first concentrate on the local linear fits.

From basic calculus, any smooth function appears linear or straight if you look at a small enough piece of the function. Thus, if we can use fits that are local enough, we should be able to fit any smooth function accurately. In practice this is not always achieved and the estimates will be biased to the extent that the function does not look like a straight line, or linear, over the region of a local linear fit. The bias will be greatest in regions of high curvature and at discontinuities in the function. In boundary regions we simply have less nearby data for our linear least-squares fits which results in more noise but does not introduce any additional bias.

In order to apply this technique to our problem we need to transform this regression method into a density estimation method. The standard way to accomplish this is by first performing a histogram on the data (e.g., [16, p. 50]). Each histogram value is then an estimate of the density function at the bin center, and we can apply a regression method to the histogrammed data. This transformation of density estimation problem into a regression problem is illustrated in Figure 4.6. This still leave us with the problem of how to form the histogram bins and how many to use. Using too many bins results in excessive computational costs while using too few can introduce additional bias due to loss of spatial information. This error is particularly a problem because it is difficult to analyze.

Instead we have devised a way to eliminate the histogramming step by taking the limit as the number of histogram bins goes to infinity. We call this method *local polynomial*

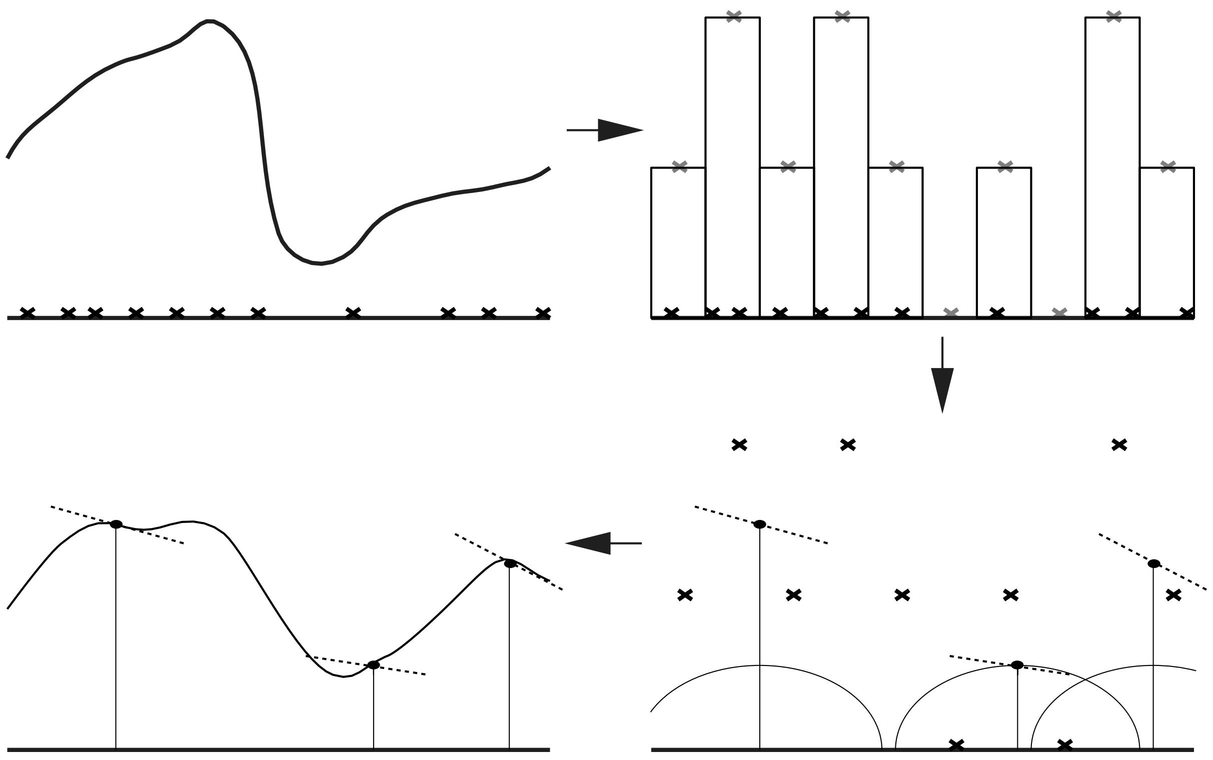

Figure 4.6: Density estimation performed using locally-weighted linear regression with a finite number of bins. Samples from a density function (top left) are histogrammed (top right),and then local linear regression is applied (bottom right) to produce the estimated density function (bottom left).

density estimation. The derivation of the local linear case is presented in Section 4.3. One surprising result is that local linear density estimation is identical to kernel density estimation in the interior, and thus we can still use the intuition we have gained about kernel density estimation. However in boundary regions this new method automatically adapts to eliminate boundary bias.

4.3 Local Linear Density Estimation

The local linear density estimation method is derived from locally-weighted linear least-squares regression. In a regression problem we are given a set of points $\{\mathbf{x}_i\}$ and corresponding, possibly noisy, estimates $\{Y_i\}$ of a function at these points. Our goal is to use these estimates to predict the function at any point. This is similiar to density estimation except that in density estimation we only have the points $\{\mathbf{X}_i\}$, not the estimates, and we have to try to reconstruct the function from the density of these points. In regression there need not be any relation between the function and the density of the sample points. We can transform a density estimation problem into a regression problem by dividing the domain into a large number of small bins and performing a histogram on the data. We can then use the bin centers as our points $\{\mathbf{X}_i\}$ and the histogram values give us our $\{Y_i\}$ estimates of the density function. The histogram value Y_i is just the number of data points in bin i divided by its area A_i and divided by the total number of data points n. In itself this is not a very good approximation for the function due to its high variability and lack of smoothness, but we can use this as input for a regression method such as locally-weighted linear least-squares regression, as illustrated for one dimension in Figure 4.6, to produce a better estimate.

We will be working in two dimensions, and will denote 2D points in boldface and their two spatial coordinates by the subscripts u and v. Let $\mathbf{x}_i$ be the center of the ith histogram bin, with area A_i and histogram value Y_i. Let m be the number of bins, and $\mathrm{K}_h(\mathbf{y})$ be the kernel function scaled for our choice of bandwidth. Then we can write the local least squares fit at point $\mathbf{x}$ as a matrix equation using the following matrices. Let B be the matrix whose columns are the basis functions for a polynomial fit, given in our case by:

$$B = \begin{bmatrix} 1 & [\mathbf{x}_1 - \mathbf{x}]_u & [\mathbf{x}_1 - \mathbf{x}]_v \\ \vdots & \vdots & \vdots \\ 1 & [\mathbf{x}_m - \mathbf{x}]_u & [\mathbf{x}_m - \mathbf{x}]_v \end{bmatrix} \tag{4.8}$$

Let W be a diagonal matrix of weights:

$$W = \begin{bmatrix} \mathrm{K}_h(\mathbf{x}_1 - \mathbf{x})A_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \mathrm{K}_h(\mathbf{x}_m - \mathbf{x})A_m \end{bmatrix} \tag{4.9}$$

and β be the vector of coefficients for the linear polynomial:

$$\beta = \begin{bmatrix} \beta_0 \\ \beta_u \\ \beta_v \end{bmatrix} \tag{4.10}$$

where the fitted polynomial is $\beta_0 + \beta_u[\mathbf{y} - \mathbf{x}]_u + \beta_v[\mathbf{y} - \mathbf{x}]_v$. Note that our function estimate at this point, $\mathbf{x}$, will just be the value of β_0. Finally, let Y be the vector of computed histogram values:

$$Y = \begin{bmatrix} Y_1 \\ \vdots \\ Y_m \end{bmatrix} \tag{4.11}$$

If the histogram values happened to lie exactly on a plane, we could solve for β using the equation $B\beta = Y$, but in general this equation will have no solution. We could instead find the closest solution in the least squares sense by using the normal equation $B^T B\beta = B^T Y$, however this would be a global least squares fit. We turn this into a local least squares fit by using a weighting matrix to give more influence to nearby values. The locally-weighted least squares fit, which can also be thought of as a minimizer of the residual quantity $||W^{1/2}B\beta - W^{1/2}Y||$, is given by:

$$B^T W B\beta = B^T W Y \tag{4.12}$$

We can multiply these matrices to find:

$$B^T W B = \begin{bmatrix} Q_{0,0} & Q_{1,0} & Q_{0,1} \\ Q_{1,0} & Q_{2,0} & Q_{1,1} \\ Q_{0,1} & Q_{1,1} & Q_{0,2} \end{bmatrix}$$

$$Q_{i,j} = \sum_{k}^{m} \mathrm{K}_h(\mathbf{x}_k - \mathbf{x}) A_i ([\mathbf{x}_k - \mathbf{x}]_u)^i ([\mathbf{x}_k - \mathbf{x}]_v)^j \tag{4.13}$$

and:

$$B^T W Y = \begin{bmatrix} \sum_i^m \mathrm{K}_h(\mathbf{x}_i - \mathbf{x}) A_i Y_i \\ \sum_i^m \mathrm{K}_h(\mathbf{x}_i - \mathbf{x}) A_i Y_i [\mathbf{x}_i - \mathbf{x}]_u \\ \sum_i^m \mathrm{K}_h(\mathbf{x}_i - \mathbf{x}) A_i Y_i [\mathbf{x}_i - \mathbf{x}]_v \end{bmatrix} \tag{4.14}$$

All of these values could be calculated and the matrix equation solved by standard techniques, as is sometimes done in the statistics literature [16]. The difficulty with this is in choosing the number of histogram bins m to use. If we use too few bins, we will have thrown away too much spatial information, adding a complicated bias to our estimate. If we use too many bins, the computation expense becomes excessive.

Fortunately, there is a way to avoid this difficulty; we take the limit as the number of bins goes to infinity. In the process, our sums turn into integrals and our histogram bins become delta functions if they lie exactly on a hit point or are equal to zero otherwise.

When we take the limit as our histogram bins shrink to infinitesmal size, our matrices become:

$$B^T W B = \begin{bmatrix} M_{0,0} & M_{1,0} & M_{0,1} \\ M_{1,0} & M_{2,0} & M_{1,1} \\ M_{0,1} & M_{1,1} & M_{0,2} \end{bmatrix}$$

$$M_{i,j} = \int_D \mathrm{K}_h(\mathbf{y} - \mathbf{x}) ([\mathbf{y} - \mathbf{x}]_u)^i ([\mathbf{y} - \mathbf{x}]_v)^j d\mathbf{y} \tag{4.15}$$

and:

$$B^T W Y = \begin{bmatrix} \frac{1}{n} \sum_j^n \mathrm{K}_h(\mathbf{X}_j - \mathbf{x}) \\ \frac{1}{n} \sum_j^n \mathrm{K}_h(\mathbf{X}_j - \mathbf{x}) [\mathbf{X}_j - \mathbf{x}]_u \\ \frac{1}{n} \sum_j^n \mathrm{K}_h(\mathbf{X}_j - \mathbf{x}) [\mathbf{X}_j - \mathbf{x}]_v \end{bmatrix} \tag{4.16}$$

where n is the number of hit points, $\mathbf{X}_j$ are the original data points, and D is the intersection of the support of the kernel and the domain, which are a circle and a polygon respectively in our case.

One interesting and important case is when we are in the interior, or in other words, when the support of the kernel lies entirely within the domain. Since we use symmetric kernels, we can easily show that the off-diagonal elements of the matrix (4.15) are zero in this case. Recalling that kernels are normalized such that $\int \mathrm{K}_h(\mathbf{y})d\mathbf{y} = 1$, our estimate reduces to the simple form:

$$\tilde{f}(\mathbf{x}) = \beta_0 = \frac{1}{n}\sum_{j}^{n} \mathrm{K}_h(\mathbf{X}_j - \mathbf{x}) \tag{4.17}$$

This is exactly the same as Equation 4.2 for standard kernel density estimation. This is an useful and perhaps surprising result.

When we are in a boundary region then the matrix $B^T W B$ will generally not be diagonal and we will need to obtain a solution using Equations 4.12, 4.15, and 4.16. By taking the limit, we have transformed a regression method into a density estimation method which turns out to be identical to the standard kernel method in non-boundary regions, but automatically adapts to eliminate boundary bias in boundary regions.

4.3.1 Computing Kernel Moments

There is one major difficulty with the local linear density estimation method derived above. Solving the integrals in Equation 4.15, which we will call the *kernel moments*, is a non-trivial problem in boundary regions. We can take advantage of the fact that our surfaces are polygons to find analytic formulas for these integrals. We accomplish this by breaking the computation up into three classes: kernel moments when the support of the kernel lies completely within the polygon, modifications to those moments when an edge of the polygon crosses the support of the kernel, and modifications to the edge effects when two edges meet at a vertex within the support of the kernel (see Figure 4.7). It can be shown that these three pieces are complete in that when summed they suffice to calculate the kernel moments on any arbitrary polygon[5].

[5]Interestingly this construction has some similiarities to the polygon anti-aliasing filters used in [17].

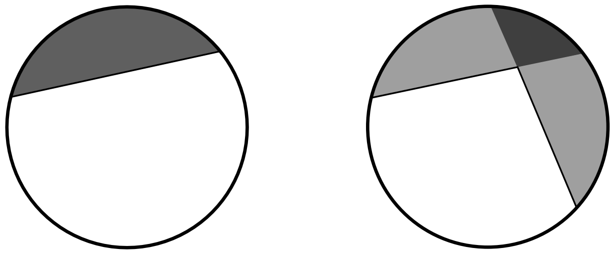

Figure 4.7: Dark shaded areas are the part of a kernel's support affected by an edge (left) and a vertex (right).

The entire kernel, edge, and vertex formulas could be solved by a symbolic math package, such as Mathematica [60]. However the results turn out to be too complicated to be useful, so we will simplify the integrals before solving them. The first simplification is to notice how Equation 4.15 changes under rotations about the evaluation point, $\mathbf{x}$. We define three quantities $\mathcal{M}_0$, $\mathcal{M}_1$, and $\mathcal{M}_2$ by:

$$B^TWB = \left[\begin{array}{c|cc} M_{0,0} & M_{1,0} & M_{0,1} \\ \hline M_{1,0} & M_{2,0} & M_{1,1} \\ M_{0,1} & M_{1,1} & M_{0,2} \end{array}\right] = \begin{bmatrix} \mathcal{M}_0 & \mathcal{M}_1 \\ {\mathcal{M}_1}^T & \mathcal{M}_2 \end{bmatrix} \tag{4.18}$$

We could work out the rotation equations directly from the integrals, but its easier to notice that $\mathcal{M}_0$, $\mathcal{M}_1$, and $\mathcal{M}_2$ are zero, first, and second order tensors [14] respectively. Given a coordinate frame and an angle θ, we can express a point, $\mathbf{y}$, in a coordinate frame rotated by θ as $\mathbf{y}' = R\mathbf{y}$ where:

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{4.19}$$

We can express the tensor moments in the rotated coordinate frame by the following formulas:

$$\begin{aligned} \mathcal{M}_0 &= \mathcal{M}'_0 \\ \mathcal{M}_1 &= \mathcal{M}'_1 R \\ \mathcal{M}_2 &= R^{-1}\mathcal{M}'_2 R \end{aligned} \tag{4.20}$$

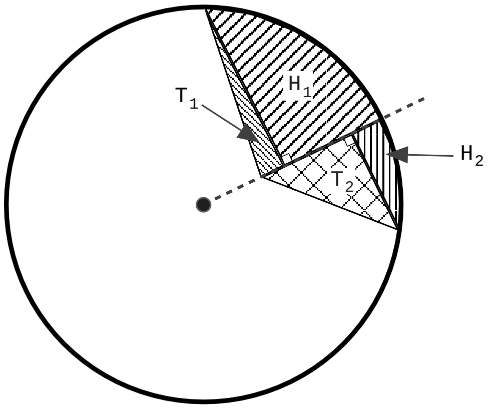

Figure 4.8: To simplify the computation vertex kernels are split into four regions: two right triangles (T_1 and T_2) and two half-edges (H_1 and H_2).

We can compute the edge and vertex moments in a canonical orientation and then rotate the result into the proper orientation using these equations. The second simplification we will use is to split the vertex moments into four pieces as shown in Figure 4.8. With this decomposition, we need only compute the moments for four simple shapes: the whole kernel, a vertical edge, a vertical half-edge, and an axis-aligned right triangle. The last three are shown in canonical position with their relevant parameters in Figure 4.9. Note that a, b, c, and d are all signed quantities. As shown they are all positive, but some care is required to get the signs correct in other cases. Let us define the following integrals for these pieces of the kernel moments:

$$M^W_{i,j} = \int_{-h}^{h} \int_{-\sqrt{h^2-u^2}}^{\sqrt{h^2-u^2}} \mathrm{K}_h(u,v) u^i v^j \, dv \, du \tag{4.21}$$

$$M^E_{i,j} = \int_{d}^{h} \int_{-\sqrt{h^2-u^2}}^{\sqrt{h^2-u^2}} \mathrm{K}_h(u,v) u^i v^j \, dv \, du \tag{4.22}$$

$$M^H_{i,j} = \int_{c}^{h} \int_{0}^{b\sqrt{\frac{h^2-u^2}{b^2}}} \mathrm{K}_h(u,v) u^i v^j \, dv \, du \tag{4.23}$$

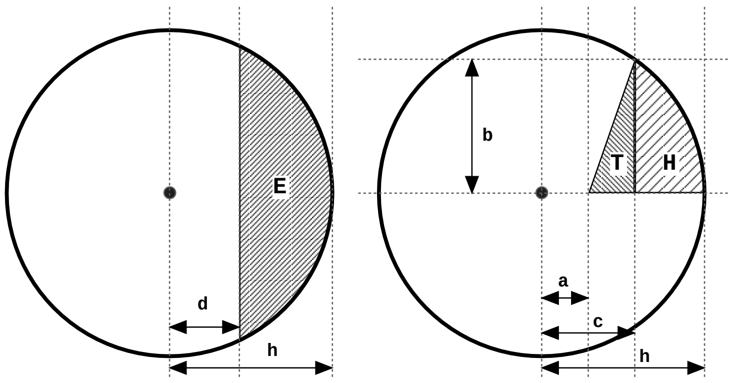

Figure 4.9: The three standard shapes for computing kernel moments: edges (E), right triangles (T), and half-edges (H).

$$M^T_{i,j} = \int_a^c \int_0^{b\frac{u-a}{c-a}} \mathrm{K}_h(u,v) u^i v^j \, dv \, du \tag{4.24}$$

If these are known, then we can handle any arbitrary polygon by rotating and summing regions of these shapes. These integrals can be solved using a symbolic math package such as Mathematica [60] or Maple. The kernel moments for the uniform kernel of Equation 4.3 are listed in Tables 4.1 - 4.4. The other kernel we commonly use is the Epanechnikov kernel whose moments are listed in Tables 4.5 - 4.8. Note these formulas are for kernels that have been rescaled by the bandwidth as in Equation 4.1.

Assembling these pieces into a complete kernel moment calculator is an exacting task, and a careful case analysis is required to get the various signs correct. We used a simple Monte Carlo integrator to check against while debugging our implementation and strongly recommend this practice. Also in some cases it is necessary to add whole kernel moments in order to get the correct result. An example is shown in Figure 4.10. These cases are easily detected since when finished, $M_{0,0}$ should always be non-negative. While computing analytic kernel moments may look difficult, once implemented we have found runtime costs to be negligible in practice. Other parts of the density estimation process dominate the computation time.

Table 4.1: Whole kernel moments for uniform kernel

$$
\begin{aligned}
M^W_{0,0} &= 1 \\
M^W_{1,0} &= 0 \\
M^W_{0,1} &= 0 \\
M^W_{2,0} &= \tfrac{h^2}{4} \\
M^W_{1,1} &= 0 \\
M^W_{0,2} &= \tfrac{h^2}{4}
\end{aligned}
$$

Table 4.2: Edge kernel moments for uniform kernel

$$
\begin{aligned}
M^E_{0,0} &= \tfrac{-de+h^2 \arccos(d/h)}{h^2\pi} \\
M^E_{1,0} &= \tfrac{2e^3}{3h^2\pi} \\
M^E_{0,1} &= 0 \\
M^E_{2,0} &= \tfrac{2de^3-deh^2+h^4 \arccos(d/h)}{4h^2\pi} \\
M^E_{1,1} &= 0 \\
M^E_{0,2} &= \tfrac{-2de^3-3deh^2+3h^4 \arccos(d/h)}{12h^2\pi} \\
e &= \sqrt{h^2-d^2}
\end{aligned}
$$

Table 4.3: Half-edge kernel moments for uniform kernel

$$\begin{aligned}
M^H_{0,0} &= \frac{-bc+h^2\frac{b}{|b|}\arccos(c/h)}{2h^2\pi} \\
M^H_{1,0} &= \frac{b^3}{3h^2\pi} \\
M^H_{0,1} &= \frac{2h^3-b^2c-2ch^2}{6h^2\pi} \\
M^H_{2,0} &= \frac{2b^3c-bch^2+h^4\frac{b}{|b|}\arccos(c/h)}{8h^2\pi} \\
M^H_{1,1} &= \frac{b^4}{8h^2\pi} \\
M^H_{0,2} &= \frac{-2b^3c-3bch^2+3h^4\frac{b}{|b|}\arccos(d/h)}{24h^2\pi}
\end{aligned}$$

Table 4.4: Triangle kernel moments for uniform kernel

$$\begin{aligned}
M^T_{0,0} &= \frac{(c-a)b}{2h^2\pi} \\
M^T_{1,0} &= \frac{b(c-a)(2c+a)}{6h^2\pi} \\
M^T_{0,1} &= \frac{b^2(c-a)}{6h^2\pi} \\
M^T_{2,0} &= \frac{b(c-a)(-3b^2+a^2+2ac+3h^2}{12h^2\pi} \\
M^T_{1,1} &= \frac{b^2(c-a)(3c+a)}{24h^2\pi} \\
M^T_{0,2} &= \frac{b^3(c-a)}{12h^2\pi}
\end{aligned}$$

Table 4.5: Whole kernel moments for Epanechnikov kernel

$$\begin{aligned}
M^W_{0,0} &= 1 \\
M^W_{1,0} &= 0 \\
M^W_{0,1} &= 0 \\
M^W_{2,0} &= \frac{h^2}{6} \\
M^W_{1,1} &= 0 \\
M^W_{0,2} &= \frac{h^2}{6}
\end{aligned}$$

Table 4.6: Edge kernel moments for Epanechnikov kernel

$$
\begin{aligned}
M^E_{0,0} &= \frac{-2de^3-3deh^2+3h^4\arccos(d/h)}{3h^4\pi}\\
M^E_{1,0} &= \frac{8e^5}{15h^4\pi}\\
M^E_{0,1} &= 0\\
M^E_{2,0} &= \frac{8de^5-2de^3h^2-3deh^4+3h^6\arccos(d/h)}{18h^4\pi}\\
M^E_{1,1} &= 0\\
M^E_{0,2} &= \frac{-8de^5-10de^3h^2-15deh^4+15h^6\arccos(d/h)}{90h^4\pi}\\
e &= \sqrt{h^2-d^2}
\end{aligned}
$$

Table 4.7: Half-edge kernel moments for Epanechnikov kernel

$$
\begin{aligned}
M^H_{0,0} &= \frac{-2b^3c-3bch^2+3h^4\frac{b}{|b|}\arccos(c/h)}{6h^4\pi}\\
M^H_{1,0} &= \frac{4b^5}{15h^4\pi}\\
M^H_{0,1} &= \frac{-3b^4c-4b^2ch^2-8ch^4+8h^5}{30h^4\pi}\\
M^H_{2,0} &= \frac{8b^5c-2b^3ch^2-3bch^4+3h^6\frac{b}{|b|}\arccos(c/h)}{36h^4\pi}\\
M^H_{1,1} &= \frac{b^6}{12h^4\pi}\\
M^H_{0,2} &= \frac{-8b^5c-10b^3ch^2-15bch^4+15h^6\frac{b}{|b|}\arccos(c/h)}{180h^4\pi}
\end{aligned}
$$

Table 4.8: Triangle kernel moments for Epanechnikov kernel

$$
\begin{aligned}
M^T_{0,0} &= \frac{b(c-a)(-a^2-2ac+2b^2+3h^2)}{6h^4\pi} \\
M^T_{1,0} &= \frac{b(c-a)(-3a^3-6a^2c+8ab^2+8cb^2+ah^2+8ch^2)}{30h^4\pi} \\
M^T_{0,1} &= \frac{b^2(c-a)(-a^2-3ac+3b^2+4h^2)}{30h^4\pi} \\
M^T_{2,0} &= \frac{b(c-a)(-6a^4-12a^3c+17a^2b^2+20ab^2c-20b^4-3a^2h^2+6ach^2+5b^2h^2)}{90h^4\pi} \\
M^T_{1,1} &= \frac{b^2(c-a)(-a^3-3a^2c+5ab^2+5b^2c+ah^2+5ch^2)}{60h^4\pi} \\
M^T_{0,2} &= \frac{b^3(c-a)(-a^2-4ac+4b^2+5h^2)}{90h^4\pi}
\end{aligned}
$$

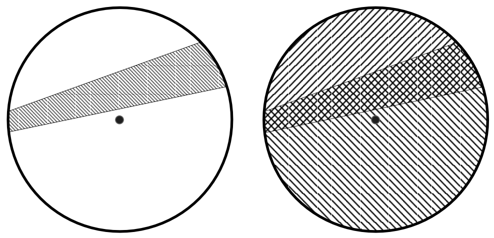

Figure 4.10: The shaded region on the left is a region that is outside the domain, or polygon, but inside the support of the kernel. When we naively include the effects of these edges we get the figure on the right, where the singly shaded regions have been subtracted once and the double shaded region have been subtracted twice. We can get back to the correct configuration on the left by simply adding back the entire area of support for the kernel once.

4.4 Local Polynomial Density Estimation

One can alternatively locally fit different functions such as constant or quadratic polynomials and the derivation remains essentially the same, although higher order fits would require many more kernel moments and are considerably more difficult to implement. One can also fit non-polynomial functions if other types of functions were desirable for some reason.

The only other function fitting that we will consider here is the local constant method which locally fits a constant function at each point. Its derivation is similar and simpler than the linear case. Before we give the resulting density estimation equation, let us first revisit the distinction between regression and density estimation problems.

4.4.1 Weighted Points in Density Estimation

In Section 4.2.1, we presented regression as the problem when data points have values associated with them and density estimation as the problem when only the density of the points is relevant. However, there is no reason that we cannot perform density estimation on points which have weights associated with them. The real distinction between density estimation and regression is whether the density of the sample points is an intrinsic part of the information we are trying to recover, or just a nuisance function that ideally would not affect our results.

Consider estimating the local population density from a random phone survey. It would be essential to ask how many people live in each household reached. A household of five could then be considered to be five data points at the same geographic location or as a single data point with five times the weight of single person household. Care is required to make sure the overall volume of the function stays the same, but in principle the results should be identical either way.

It is possible to preprocess the data to force each data point to have the same weight, but this can result in a loss of information. It is often preferable to allow each data point to have an associated weight w_i. The summarized density estimation equations in the next section will include these weighting factors.

4.4.2 Naive, Constant, and Linear Density Estimation Revisited

The three density estimation methods that we have discussed so far are the standard, or naive, kernel density estimation and the local constant and linear methods. It will be useful to restate their equations here for the data points with weights.

The naive method is given by:

$$\tilde{f}(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} w_i \, \mathrm{K}_h(\mathbf{x} - \mathbf{X}_i) \tag{4.25}$$

The local constant method is given by:

$$\tilde{f}(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} w_i \, \frac{\mathrm{K}_h(\mathbf{x} - \mathbf{X}_i)}{M_{0,0}} \tag{4.26}$$

where $M_{0,0}$ is a kernel moment as defined in Equation 4.15. It is simply the volume of the kernel function and is equal to one in the interior.

The local linear method is given by:

$$\tilde{f}(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} w_i \left(\left\{ \hat{\mathbf{e}}_1^T \cdot \left(B^T W B\right)^{-1} \right\} \cdot \begin{bmatrix} 1 \\ [\mathbf{X}_i - \mathbf{x}]_u \\ [\mathbf{X}_i - \mathbf{x}]_v \end{bmatrix} \right) \mathrm{K}_h(\mathbf{x} - \mathbf{X}_i) \tag{4.27}$$

where $\hat{\mathbf{e}}_1 = [1,\, 0,\, 0]^T$ and we take its dot product with the inverse of the kernel moment matrix $B^T W B$, which was defined in Equation 4.15. Thus the quantity in curly braces is a row vector that does not depend on i.

The most expensive part for all these density estimation is computing the summation over the data points. The local constant method has very little incremental cost over the naive method since their summations are the same, but the local linear method is somewhat more expensive. In the interior all three methods produce identical results, so if we can detect when we are in the interior we can always use the cheaper naive method. This turns out to be trivial. As part of computing the kernel moments we check to see if any edges cross the support of the kernel. If none do, then we are in the interior and use the naive method. Thus we need only pay the extra cost of the local linear method in boundary regions where it is of benefit.

In general we recommend using the local linear method for its superior boundary bias behavior, but there are valid reasons for sometimes using the local constant method. The local linear method is biased to the extent the function does not look locally like a constant gradient (e.g., curvature or discontinuities cause bias). In the interior, where all three methods produce identical results, all three methods share this property. In boundary regions the local constant method is unbiased as long as the function looks locally constant (i.e. gradients cause bias in boundary regions, but not in the interior). The naive method, on the other hand, is biased in boundary regions for all functions. This makes the naive method unacceptable for any application where the boundary region is important.

Compared to the naive method, the asymptotic boundary bias of the local constant method is much less objectionable. Its relative simplicity compared to the local linear method makes it preferable in some cases. Whenever boundary bias is likely to be important or noticeable, then the local linear method is the method of choice. One common example is when a surface that should appear continuous was been split in the modeling process. Without the linear method, differing boundary biases on either side of the split can cause a jump discontinuity in the lighting across the split that is quite noticeable and objectionable[6].

Higher order fits such as local quadratic would tend to perform better in smooth

[6]Note there will also be a discontinuity in the solution noise across the boundary. But since our bandwidths are chosen, in part, to reduce the noise to an imperceptible level, the noise induced part of the discontinuity will hopefully not be visible.

regions, but worse around discontinuities and other sharp illumination features such as shadow boundaries. Higher order fits generally require larger bandwidths to achieve the same variance level, but near discontinuities, using smaller bandwidths is the only truly effective way to reduce the bias. They also can cause perceptually objecctionable overshoot, or "ringing", artifacts near sharp features.

4.4.3 Equivalent Kernels

Most kernel-based density estimation methods, including the local polynomial methods, can be written in a form similiar to the naive method (Equation 4.25) by folding the bias correction terms into the kernel function. Such modified kernels are known as equivalent kernels and their shape is no longer independent of the evaluation point. We write such kernels as $\mathrm{K}'_{h,\mathbf{x}}$ to emphasize their dependence on the evaluation point $\mathbf{x}$. It is straightforward to rewrite the naive, local constant, and local linear methods (Equations 4.25, 4.26, and 4.27) in the form:

$$\tilde{f}(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^{n} w_i \, \mathrm{K}'_{h,\mathbf{x}}(\mathbf{x} - \mathbf{X}_i) \tag{4.28}$$

For most users, equivalent kernels are not a particularly useful way to think about local polynomial methods, but they are discussed and used in the statistics literature (e.g., [25]) because they provide a way to compare different boundary bias schemes. Using equivalent kernels, one can consider the local polynomial density estimation methods as being simply another variant of boundary kernels. The real advantage of local polynomial methods is not that their results could not be duplicated by older, more ad hoc methods, but rather that they provide a clean conceptual framework for dealing with boundary regions and boundary bias.

In the next two chapters we will be discussing various methods to try to extract as much information from the particle data as possible. For simplicity, the equations guiding our decisions will be written in terms of the naive method. However except where noted, all these equations can be extended to handle the local polynomial boundary bias correction methods by substituting the appropiate equivalent kernel in place of the kernel of the naive method.

Chapter 5

Bandwidth Selection

In this chapter we return to the important issue of choosing bandwidths for kernel-based density estimation methods. This was briefly described in Section 4.1.2. Recall that the bandwidth is the name given to the parameter h that controls the width of the kernels via Equation 4.1, and determines the size of the local regions that we use. The bandwidth selection described here can be used with any of the boundary bias correction techniques from the previous chapter.

The most straightforward way to get better density estimation results is to use more particles to generate more data, but unfortunately the particle tracing is already the most expensive part of our computations. For a fixed amount of data, the bandwidth is the single most important parameter. It determines a tradeoff between two types of error: bias and noise, as illustrated in Figure 5.1. Our goal in this chapter is to dynamically optimize this tradeoff using spatially adaptive bandwidths by developing a good automatic bandwidth selector.

5.1 Bandwidth Selection Strategies

Many different bandwidth selection strategies and rationales have been proposed. The most popular bandwidth selection techniques among graphics researchers (e.g., [31, 10, 51]) have been variations of the k-th nearest neighbor method. The bandwidth is chosen to be just large enough that the kernel contains k data points, where k is generally a user specified parameter. If the data points all have the same weight w_i in the sense of Section 4.4.2, then this results in a constant relative noise level[1] (e.g., 2%).

This is the approach we used in our initial work [47, 56]. The bandwidth was constant over each individual surface and chosen so that on average a user-specified number of data points fall within each kernel. Setting this parameter forced the user into the unsatisfactory dilemma illustrated in Figure 5.1. Setting it large enough to eliminate all noise caused objectionable blurring of important features. Smaller settings resulted in distracting visible noise artifacts that most users also found equally objectionable.

From these experiences, it was clear to us that using spatially adaptive bandwidths near features could produce much better results. Because we typically evaluate the density at millions of points, this requires an automatic inexpensive bandwidth selector, although some additional cost is inevitable and justified by superior results. Ideally the selector should include both bias and noise considerations and not require user intervention or tuning.

5.1.1 Mean Squared Error Metric

The standard approach in the density estimation literature[32, 57] is to pick bandwidths to try to minimize the expected value of the squared deviation between the density estimate and the true density. At a single point this error metric is called the mean squared error

[1]This is a desirable property as we will discuss in relation to Weber's law, but we want a technique that can achieve this property even when particles are unequally weighted as ours are.

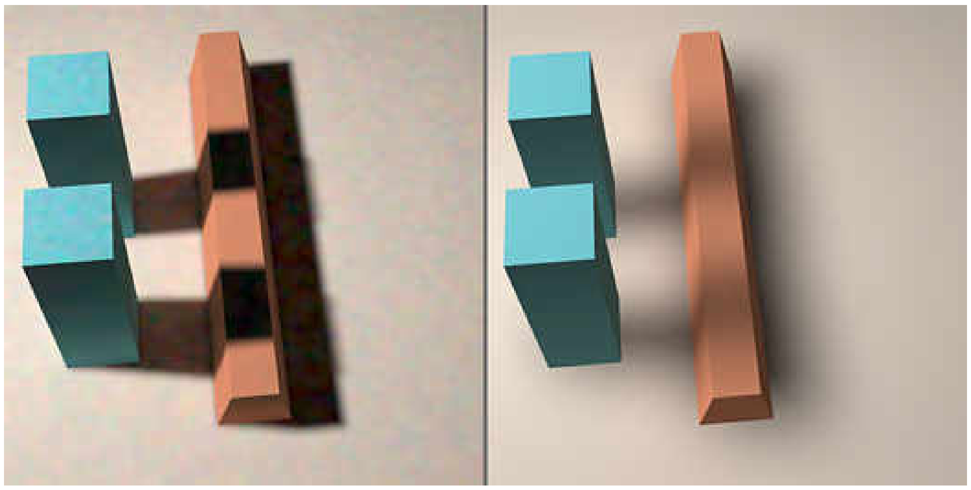

Figure 5.1: Two solutions using a simple fixed bandwidth kernel density estimation. Too small a bandwidth causes excessive noise (left) while too large a bandwidth causes excessive bias or blurring(right). A better idea is to use spatially adaptive bandwidths to achieve both low noise and good feature resolution.

(MSE). When summed over a region it is referred to as the mean integrated squared error (MISE).

Squared error metrics, such as the MSE, are appealing because they are easier to analyze than most other error metrics. In many problems, analytic formulas can be derived for finding their minimizers. Unfortunately, this is not true for nonparametric density estimation where the MSE can only be evaluated when the density function is known or in the asymptotic limit of infinite data. In practice the mean squared error and its optimal bandwidth can only be estimated heuristically. The mean squared error of the estimate $\tilde{f}$ is equal to the squared bias plus the variance:

$$\mathrm{MSE}\,\tilde{f}(\mathbf{x}) \;=\; \mathrm{E}\left(\tilde{f}(\mathbf{x}) - f(\mathbf{x})\right)^2 \;=\; \left(\mathrm{E}\,\tilde{f}(\mathbf{x}) - f(\mathbf{x})\right)^2 + \mathrm{Var}\,\tilde{f}(\mathbf{x}) \tag{5.1}$$

We could further expand this using Equations 4.5 and 4.6, but it will always remain in terms of the unknown density function f. The variance component is relatively easy to estimate, but the bias is not. This is unfortunate as, unlike the variance, the bias could easily be removed if it was accurately known.

As shown in Equation 4.5, in kernel-based density estimation methods, the expected value of the estimator is equal to a blurred version of the true density function. Thus the bias is equal to the difference between a blurred version and the true function. As such it depends sensitively on the presence or absence of small scale or high frequency features in the true function. These are precisely the components of the density function about which we have the least information and which are most difficult to estimate.

Methods which attempt to minimize the mean squared error have to use some heuristic to estimate the MSE. For example Myszkowski [39] has published results based on our first version of the density estimation framework [47] but modified in several ways including using variable kernel widths. Although his bandwidth selection theory is based on minimizing the mean squared error, he decides that an accurate MSE estimator would be too difficult and expensive. Instead he uses a simple heuristic based on differences of local estimates. His heuristic is frequently, but not always, correlated to the true mean squared error. For example, his heuristic considers a local gradient to be bias inducing, but only elements beyond a constant gradient, such as curvature, actually cause bias[2].

It is clear that adaptive bandwidths are a good idea, but our bandwidth selector need not be based on mean square error. The MSE is hard to evaluate and may not be the same as the application specific error that we really want to minimize. For example zooming into Myszkowski's results reveals that they still contain potentially visible noise artifacts. For these reasons, we feel well justified in looking at other rationales for choosing bandwidths.

[2]This is true in the interior for all symmetric kernel-based methods and true even in boundary regions for the local linear method as we discussed in the previous chapter.

5.1.2 Perceptual Error

We expect that, in most cases, our results will be displayed to a human observer in a realistic manner (i.e. in a way intended to mimic real world appearances). In this case, the relevant error is how closely does the observer's perception of our results resemble what their perception would be of the correct results. Unfortunately, perceptual errors are hard to model, quantify, or measure. Current models of human visual perception are still evolving [15], but some graphics researchers (e.g., [18, 9]) have tried to apply these models. For us, the complexity of these models presents one difficulty, but the real obstacle is that we do not and can not know some of the information required by the more complete models. Instead we will use some relatively simplistic standard perceptual procedures such as Weber's law and CIE $L^*a^*b^*$ space.

The density estimation framework produces view-independent results, but what the user actually sees are images on an output device, typically a computer monitor. The mapping from our results to images, which consists of choice of view and tone mapping, will be under user control and hence is necessarily unknown to us. The view controls which surfaces will be visible and how they will be spatially warped in the mapping. This means that there are important perceptual factors such as the resultant spatial frequencies on the display that we cannot know at computation time. To circumvent this problem, we use conservative perceptual metrics that can assure us that a pattern will remain imperceptible regardless of spatial warping.

Most display devices are only capable of reproducing a small fraction of the range of tristimulus values present in our results. The problem of mapping a larger set of tristimulus values to the limited gamut of a display device is known as tone mapping. The tone mapping will also be under user control, but we need to make some assumptions about it to be able to proceed. As a working basis, we assume that the tone mapping is essentially compressive in nature, or more specifically that chromaticities are preserved and that the relative luminance differences are not exaggerated.

5.2 Basic Approach

Our goal in selecting bandwidths is to preserve the important features in illumination reconstruction while suppressing spurious perceptual artifacts; the latter being primarily caused by noise in our data. We begin our bandwidth selection process by using some metrics for measuring the visibility of noise. Our estimations of a color stimuli functions are split into separate estimations of luminous and chromatic parts, for which we use perceptual noise metrics based on Weber's law and the CIE $L^*a^*b^*$ color standard respectively. These noise metrics are somewhat simplistic and conservative, but are also relatively easy to compute and use. They do not account for spatial frequency, but our intention is to insure that color stimuli are imperceptibly different regardless of the spatial pattern in which they appear. We will use these metrics to choose initial bandwidths to insure that the remaining noise will not be visible.

In some cases however, reducing the noise to such a low level may cause an excessive amount of bias. This causes undesirable blurring of important features such as the edges of shadows and caustics (e.g., on the right in Figure 5.1). In general the way to reduce the bias is to reduce the bandwidth (see Figure 5.2), but this may also reintroduce visible noise artifacts. Only in some cases will the reduction in bias be perceptually significant enough to justify the consequent increase in noise.

To address this concern, we introduce a bias detection technique based on a statistical model of our solution process. It automatically searches for cases where a significant change in the bias can be reliably detected above the inherent background noise and a reduction in bandwidth is perceptually justified. One nice feature of this bias detection technique is that it enhances the resolution of important illumination features without regard to their underlying physical causes (i.e. it could be a shadow, caustic, or any other illumination feature).

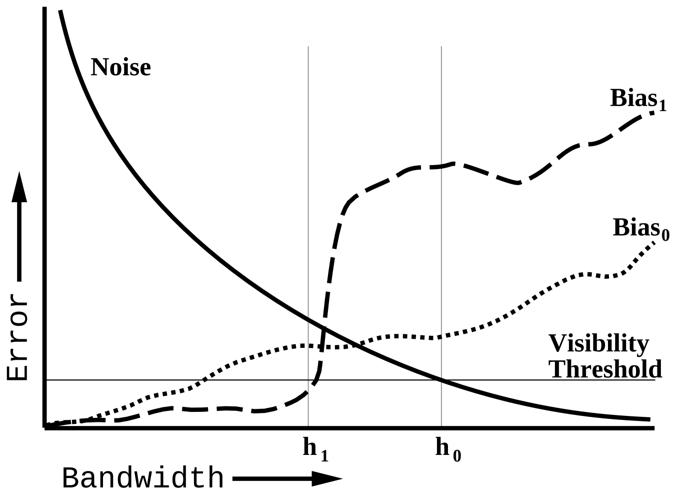

Figure 5.2: Bias and noise as functions of the bandwidth. Two different example bias curves are shown. In most cases the the bias is a slowly varying function of the bandwidth (short dashes). In this case we choose the bandwidth (h_0) to eliminate the visible noise. However near a sharp feature, the bias shows a marked increase when the kernel crosses the feature boundary (long dashes). Then we want to choose a bandwidth (h_1) to avoid blurring this feature.

5.2.1 Luminance and Chromaticity

The most obvious artifacts in Figure 5.1 are the chromatic noise and the blurring of the shadow which is primarily a brightness, or luminance, feature. By splitting the estimation process into separate estimates of the luminous and chromatic components of a color, we can apply different amounts of blurring to each and optimize the bias vs. noise tradeoff in each component individually. This split is similar to many common color encodings for compression such as in NTSC television. Rapid variations are more common and perceptually important in the luminance components than in the chromatic components. Hence it is generally in the luminance channel Y where excessive bias or blurring is most objectionable, and currently we do not use bias detection in our chromatic estimation.

Naively combining estimates at different bandwidths may cause perceptual artifacts due to different amounts of bias. These artifacts can be minimized by combining the luminance computed at one bandwidth with the chromaticity computed at a different bandwidth. Because the chromaticity (Equation 2.10) is invariant under changes in overall intensity level, and this prevents changes in bias due to differing bandwidths from causing anomalous color shifts. If Y_0 is our luminous estimate at one bandwidth and X_1, Y_1, and Z_1 are the tristimulus values estimated at a different bandwidth, then converting to chromaticity, combining them, and converting back to tristimulus values gives us:

$$\mathrm{X} = \frac{\mathrm{X}_1 \mathrm{Y}_0}{\mathrm{Y}_1}, \quad \mathrm{Y} = \mathrm{Y}_0, \quad \mathrm{Z} = \frac{\mathrm{Z}_1 \mathrm{Y}_0}{\mathrm{Y}_1} \tag{5.2}$$

We will actually compute the tristimulus exitance (see Equation 2.14), but the combination of its tristimulus channels is the same as given here.

5.2.2 Noise and Just-Noticeable-Differences

Humans have limited ability to distinguish two similar stimuli. The minimum deviation from a base stimulus which is needed for the change to be noticeable is called a just-noticeable-difference (JND) [61]. Our initial goal is to reduce the noise level to just below a JND. There would be little advantage in reducing the noise any further.

There are many different models for predicting visual JNDs [61]. Unfortunately, most depend on factors that we are assuming to be unknown. We will use Weber's law, which is one of the earliest and simplest method for estimating luminance JNDs [8]. It states that a luminance JND is approximately a fixed fraction of the base luminance.

For chromaticity JNDs, we will derive a simple metric from the CIE $\mathrm{L}^*\mathrm{a}^*\mathrm{b}^*$ color space and color difference standards. In order to measure our chromaticity noise, we will first linearly transform from XYZ space to a color opponent space based on perceptual experiments [28, 43]. We then use a Taylor expansion to derive a local approximation to the CIE $\mathrm{L}^*\mathrm{a}^*\mathrm{b}^*$ transform and use a conservative approximation to the CIE 1994 color difference formula to measure the visibility of our chromaticity noise.

5.2.3 Bias Considerations

Controlling the visible noise is sufficient in many regions, but near important features such as shadow boundaries and caustics, controlling the bias may be a more important consideration. It is often desirable to reduce the bandwidth near such features to improve their resolution even if some noise may be visible. There is also a beneficial tendency of strong features to mask nearby noise that would otherwise be visible [18].

Instead of directly estimating the bias, which is quite difficult, we have devised a simple statistical procedure to identify regions of high bias without needing to explicitly estimate the bias. The procedure first computes multiple estimates at a point using different bandwidths. Any deviation between the estimates can be due partially to noise, but also includes any change in the bias as a function of the bandwidth. Using a statistical model of our solution process, we estimate the expected noise level and determine when a deviation is sufficiently unlikely to have been caused by noise alone. When a change in the bias is detected, we use the estimate at the smaller bandwidth, as it will generally contain less bias. Ideally, we should have a perceptual criteria for determining how significant the detected bias is, but in practice we find that whenever the bias is large enough to be detected, it is also large enough to justify reducing the bandwidth.

Together these techniques provide a perceptually motivated way to automatically and adaptively select bandwidths to produce better results.

5.3 Perceptually-Motivated Bandwidth Selector

Recall that kernel density estimation works by blurring the input data points to produce a smooth estimate of the density function. The blurring is useful in that it filters out much of the noise that is inherent in the data; however it also tends to blur out genuine features in the input data. In order to understand and control this tradeoff, we need to understand the equations that govern the kernel density estimation, our perceptual JND metrics, and our bias detection. For simplicity we will write the equations for the naive kernel method, but the method is equally applicable to the local polynomial methods listed in Section 4.4.2.

The particle tracing phase produces a set of n data points, each of which consists of a location $\mathbf{X}_i$, and a wavelength λ_i. It is designed so that the probability density function p of the data points is related to the spectral irradiance E_λ, as follows:

$$p(\mathbf{x}, \lambda) = \frac{E_\lambda(\mathbf{x}, \lambda)}{V} \tag{5.3}$$

where V is the total volume of the spectral irradiance, or in other words the total light power striking all surfaces. It is easily approximated as the sum of all the powers associated with all the hit points. Our goal is to recover functions such as the illuminance and tristimulus exitance which can be written in the form:

$$f(\mathbf{x}) = \int_0^\infty E_\lambda(\mathbf{x}, \lambda)\, r(\lambda)\, d\lambda \tag{5.4}$$

where r is the corresponding spectral response curve for the function of interest.

Our estimator of the function f is:

$$\tilde{f}(\mathbf{x}; h) = \sum_{i=1}^{n} \frac{\tilde{f}_i(\mathbf{x}; h)}{n} = \sum_{i=1}^{n} \frac{V}{n}\, \mathrm{K}_h(\mathbf{x} - \mathbf{X}_i)\, r(\lambda_i) \tag{5.5}$$

where $\tilde{f}_i$ is the estimator using only the ith data point and our estimator is an average using all the data points. Using the probability density of the data points (Equation 5.3), we can find that the expected value of this estimator is:

$$\begin{aligned} \mathrm{E}\, \tilde{f}(\mathbf{x}; h) &= \int_{\mathcal{S}} \int_0^\infty V\, \mathrm{K}_h(\mathbf{x} - \mathbf{y})\, r(\lambda)\, p(\mathbf{y}, \lambda)\, d\lambda\, d\mathbf{y} \\ &= \int_{\mathcal{S}} \mathrm{K}_h(\mathbf{x} - \mathbf{y})\, f(\mathbf{y})\, d\mathbf{y} \end{aligned} \tag{5.6}$$

where $\mathcal{S}$ is the current surface.

The noise in our estimator can be characterized by its variance, which we will denote σ^2.

$$\sigma^2(\mathbf{x}; h) = \mathrm{E}\, \tilde{f}^2(\mathbf{x}; h) - \left(\mathrm{E}\, \tilde{f}(\mathbf{x}; h)\right)^2 \tag{5.7}$$

It is necessary to make a few approximations before estimating the variance. First we will assume the data points are approximately independent and identically distributed (commonly abbreviated as *iid*)[3]. The variance of a sum of *iid* components is equal to the sum of the variances of the components.

$$\sigma^2(\mathbf{x};h) \approx \sum_{i=1}^{n} \left\{ \mathrm{E}\left(\frac{\tilde{f}_i(\mathbf{x};h)}{n}\right)^2 - \left(\mathrm{E}\,\frac{\tilde{f}_i(\mathbf{x};h)}{n}\right)^2 \right\} \tag{5.8}$$

We next neglect the second term in this equation. Its omission can only increase our estimate of the variance, and we expect it to be negligible anyway. Evaluating the expected value, we now have:

$$\sigma^2(\mathbf{x};h) \approx n \int_{\mathcal{S}} \int_0^\infty \frac{V^2}{n^2}\, \mathrm{K}_h^2(\mathbf{x}-\mathbf{y})\, r^2(\lambda)\, p(\mathbf{y},\lambda)\, d\lambda\, d\mathbf{y} \tag{5.9}$$

We still cannot evaluate this equation analytically as the probability function p is assumed to be unknown, but we can easily create an estimator $\tilde{\sigma}^2$ with the appropriate expected value.

$$\tilde{\sigma}^2(\mathbf{x};h) = \frac{V^2}{n^2} \sum_{i=1}^{n} \mathrm{K}_h^2(\mathbf{x}-\mathbf{X}_i)\, r^2(\lambda_i) \tag{5.10}$$

We will pick our initial bandwidths to achieve a particular target standard deviation σ, which is just the square root of the variance σ^2. The standard deviation tells us how large the random deviations due to noise are likely to be. One of the Central Limit Theorems states that the random error in an *iid* sum converges towards a gaussian, or normal, distribution as the number of terms increases. We can use this approximation to choose a target σ such that the noise is unlikely to exceed a visibility threshold. For example, under a normal distribution the chance of a random deviation being larger than the standard deviation σ, is about 1 in 3 whereas the chance of seeing a deviation greater than 4σ is less than 1 in 10,000.

[3]The particles are *iid*, but the data points caused by a single particle are correlated to some extent. As more particles are used the significance of this correlation decreases. For large numbers of particles, it is not usually noticeable outside of artificial pathological cases.

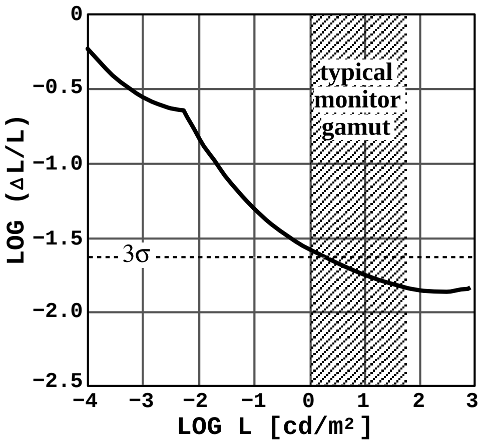

Figure 5.3: Measured luminance just-noticeable-differences ΔL, as a function of the base luminance L [61, p. 569]. Only values within the gamut of our output device (shaded region) are of interest to us. The region below the 3σ line shows the common range of noise values we expect using using our empirical value of $C_{\mathrm{Y}} = 0.008$.

5.3.1 Estimating Luminances

Let us begin by estimating the Y channel of the tristimulus exitance $M_{\mathrm{t},Y}$, which is the same as the luminous exitance M_{v}. Recall that during the reconstruction phase (but not the transport phase), we approximate all surfaces as being Lambertian. Using Equations 2.5, 2.14, and 2.24, the luminous exitance at a Lambertian surface is equal to:

$$M_{\mathrm{t},Y}(\mathbf{x}) = 683 \int_0^\infty \rho(\lambda)\, \bar{y}(\lambda)\, E_\lambda(\mathbf{x}, \lambda)\, d\lambda \tag{5.11}$$

Since our hit points are distributed according to the spectral irradiance, we can estimate the luminous exitance by setting r in Equation 5.5 equal to:

$$r_{\mathrm{Y}}(\lambda) = 683\, \rho(\lambda)\, \bar{y}(\lambda) \tag{5.12}$$

Our goal is to find bandwidths such that the noise in our luminance estimates is sufficiently unlikely to exceed a just-noticeable-difference. According to Weber's law, a JND is a fixed percentage of the base luminance, which would correspond to a horizontal line in Figure 5.3. In general this does not agree with the measured JNDs, but may serve as a reasonable approximation within the limited gamut of a typical computer monitor. We choose a bandwidth such that our Weber's law noise metric is equal to some visibility threshold value C_{Y}.

$$\frac{\tilde{\sigma}(\mathbf{x};h)}{\tilde{f}(\mathbf{x};h)} = C_{\mathrm{Y}} \tag{5.13}$$

From our own experiences we have empirically determined $C_{\mathrm{Y}} = 0.008$ as a reasonable value at which the noise is only rarely visible. Figure 5.3 shows a plot of luminous JNDs from the psychophysical literature [61]. These are measured by placing a large spot of one luminance on a larger region of a background luminance and measuring how large the difference in luminances must be for test subjects to be able to perceive them. Because of the sharp boundary between the two regions, these tests represent a worst case spatial pattern. Sharper boundaries are almost easier to perceive. If we choose 3σ as the range of typical common noise values, we can see in Figure 5.3 that our value is slightly higher than would be expected from psychophysical data. We suspect this is because although our noise can have a high apparent spatial frequency, it is smooth and thus less noticeable than patterns used in these psychophysical experiments.

Since all these quantities contain some noise, we only require that Equation 5.13 hold approximately (e.g., $\pm 5\%$)[4], but finding a suitable bandwidth is still a nontrivial task that involves a numerical search. In the worst case, the search sometimes requires twenty or more kernel evaluations at different bandwidths.

We have been able to reduce this cost by using a hierarchical refinement technique. We store our view-independent results as a piece-wise linear, or Gouraud shaded, triangle mesh on each surface. The density estimation process that we are discussing is used to find the color at each vertex. We construct this mesh iteratively, first estimating with a coarse mesh. We then repeatedly refine the mesh until the vertices are spaced less than

[4]If this tolerance is too small, the numerical search may never terminate. If it is too large, random fluctuations in the bandwidth can become visible due to changes in the bias. In our experience however, this tolerance need not be chosen very precisely.

half a bandwidth apart. During mesh refinements we interpolate the bandwidths from previous calculations and use these as initial guesses in the bandwidth search. Because the bandwidths are usually slowly varying, the initial guesses are often sufficient to satisfy Equation 5.13. We find that the average number of bandwidths evaluated per vertex is between 1.2 and 1.6. Since we must evaluate at least one bandwidth per vertex regardless, this means that the additional cost for this bandwidth search is quite reasonable.

One caveat is that a simple numerical procedure may not work when using the local linear boundary bias correction. In boundary regions, the local linear method increases the amount of noise in our estimate of $\tilde{\sigma}$ such that Equation 5.13 may be satisfied by significantly different bandwidths. This can cause unwanted variability in the chosen bandwidths and visible artifacts due to consequent changes in the bias. Rather than abandon our simple interpolation/extrapolation bandwidth search procedure, we always use the local constant method when finding the bandwidth. In boundary regions, we can switch to the local linear method after the "no visible noise" bandwidth has been found. This is less than ideal as the local linear method has somewhat higher variance in boundary regions, but seems to work in practice.

5.3.2 A Chromatic Metric

Next we want to take the same approach for our chromaticity estimates and find bandwidths such that our chromatic noise is slightly below a just-noticeable-difference. First we need a chromatic noise metric, which we derive from the CIE $L^*a^*b^*$ color space and color difference standards.

Because chromaticities involve a nonlinear transform from a tristimulus space, we cannot estimate them directly, but must first estimate the tristimulus values. Since the final luminance component will come from a separate estimate at a different bandwidth, we want to ensure that we measure only the chromatic part of our noise and exclude the luminance part. The easiest way to do this is to first perform a linear transform from

XYZ space to a color-opponent space as discussed in Section 2.2.2. There is still some uncertainty about the exact relation between these channels and standard XYZ space, and we have chosen to use a convenient relation previously suggested by Hurvich [28]. Our luminance channel is Y, our red vs. green channel is (X−Y), and our yellow vs. blue channel is (Y−Z).

Since there is a simple linear transform between XYZ and our color-opponent space, estimating in either space will not alter the resulting tristimulus values. It does, however, affect the way we measure the uncertainty or noise in our estimates. Since we model the opponent channels as being roughly perceptually independent, it is sufficient to measure the variance in each channel separately. Using the standard X, Y, and Z channels would require the extra computation and complexity of handling covariances between the channels.

The CIE L*a*b* color space was designed to be nearly perceptually uniform and is recommended by CIE for measuring perceptual color differences. The transform to CIE L*a*b* space written in terms of our Y, (X−Y), and (Y−Z) channels is [61]:

$$\mathrm{L}^* = 116\, g\left(\frac{\mathrm{Y}}{\mathrm{Y}_n}\right) - 16 \tag{5.14}$$

$$\mathrm{a}^* = 500\left[g\left(\frac{(\mathrm{X}-\mathrm{Y})+\mathrm{Y}}{\mathrm{X}_n}\right) - g\left(\frac{\mathrm{Y}}{\mathrm{Y}_n}\right)\right] \tag{5.15}$$

$$\mathrm{b}^* = 200\left[g\left(\frac{\mathrm{Y}}{\mathrm{Y}_n}\right) - g\left(\frac{\mathrm{Y}-(\mathrm{Y}-\mathrm{Z})}{\mathrm{Z}_n}\right)\right] \tag{5.16}$$

$$g(x) = \begin{cases} x^{1/3} & \text{if } x > 0.008856 \\ 7.787x + \frac{16}{116} & \text{if } x \leq 0.008856 \end{cases} \tag{5.17}$$

where X_n, Y_n, and Z_n are the tristimulus values of a nominally white stimulus under current viewing conditions.

Our goal is to estimate how large a deviation in CIE L*a*b* space we can expect due to noise in our estimates of the color opponent channels. We estimate these channels using the r functions from Equation 5.12 and below together with Equations 5.5 and 5.10.

$$r_{(\mathrm{X-Y})}(\lambda) = 683\,\rho(\lambda)\ \{\bar{x}(\lambda) - \bar{y}(\lambda)\} \tag{5.18}$$

$$r_{(\mathrm{Y-Z})}(\lambda) = 683\,\rho(\lambda)\ \{\bar{y}(\lambda) - \bar{z}(\lambda)\} \tag{5.19}$$

Next we transform the variances in (X–Y) and (Y–Z) into estimates of standard deviations in CIE L*a*b* space. Since we are primarily interested in small deviations near the limit of perceptibility, we will use a simple linear Taylor expansion of the CIE L*a*b* transform about our estimated color. Noting that a* does not depend on (Y−Z) and b* does not depend on (X−Y), we can approximate:

$$\tilde{\sigma}^*_{\mathrm{a}} = \left|\frac{\partial \mathrm{a}^*}{\partial(\mathrm{X-Y})}\right| \tilde{\sigma}_{(\mathrm{X-Y})} \tag{5.20}$$

$$\tilde{\sigma}^*_{\mathrm{b}} = \left|\frac{\partial \mathrm{b}^*}{\partial(\mathrm{Y-Z})}\right| \tilde{\sigma}_{(\mathrm{Y-Z})} \tag{5.21}$$

where the partial derivatives are evaluated at our estimated values for Y, (X−Y), and (Y–Z). Finally we combine these together to get our perceptual chromaticity noise metric $\tilde{\sigma}_c$:

$$\tilde{\sigma}_c = \frac{\sqrt{\tilde{\sigma}^2_{\mathrm{a}^*} + \tilde{\sigma}^2_{\mathrm{b}^*}}}{1 + 0.015\sqrt{(\mathrm{a}^*)^2 + (\mathrm{b}^*)^2}} \tag{5.22}$$

The denominator in this equation is inspired by CIE's 1994 revision of the CIE L*a*b* color difference formula [11, 15]. It was recognized that the original formulation tended to overestimate the perceptual differences among saturated colors, and a corrective factor was standardized. Their correction is somewhat complex to compute, so we have used a simpler but more conservative correction factor.

Before we can use this metric we need to select X_n, Y_n, and Z_n. Since chromatic differences are easier to see at high luminance levels, we conservatively pick Y_n to be the same as Y in our color-opponent space estimate. For lack of better information about chromatic adaptation, we assume that an equal energy, or flat, spectrum will appear as white and set $\mathrm{X}_n = \mathrm{Y}_n = \mathrm{Z}_n = \mathrm{Y}$.

With this chromatic noise metric, finding an appropriate chromatic bandwidth proceeds in essentially the same way as the search in the preceding section. We choose a threshold C_c and search for a bandwidth which approximately satifies:

$$\tilde{\sigma}_c(\mathbf{x}; h) = C_c \tag{5.23}$$

Empirically, we have found $C_c = 0.33$ as the value where the chromaticity noise is rarely visible. CIE L*a*b* space was designed so that just-noticeable-differences correspond to differences whose magnitude is near one or unity. Given our value for C_c, this corresponds to random deviations of three standard deviations or more, and our empirical value agrees quite well with the CIE specification.

5.3.3 Bias Detection

Our emphasis so far has been on measuring and eliminating perceptual noise. We also need to be concerned with the bias or blurring of features. The goal of the bias detection procedure is to identify points and bandwidths where the reduction in bias caused by a reduction in the bandwidth is more important than the consequent increase in noise.

Unlike the noise, the bias is quite difficult to estimate directly because it depends sensitively on fine details of the unknown density function. One thing we can do easily is to make estimates using two different bandwidths and look for reliable evidence of a change in the bias between these estimates. We form a new estimator β as a difference of two estimators:

$$\beta(\mathbf{x}; h_1, h_2) = \tilde{f}(\mathbf{x}; h_1) - \tilde{f}(\mathbf{x}; h_2) \tag{5.24}$$

The expected value of this estimator is:

$$\mathrm{E}\,\beta(\mathbf{x}; h_1, h_2) = \mathrm{E}\,\tilde{f}(\mathbf{x}; h_1) - \mathrm{E}\,\tilde{f}(\mathbf{x}; h_2) \tag{5.25}$$

which is equal to the change in the bias between the two component estimates. Even if there is no change in the bias between these two bandwidths, its actual value is unlikely to be zero due to noise.

The variance σ_β^2 of β is:

$$\sigma_\beta^2(\mathbf{x}; h_1, h_2) = \mathrm{E}\,\beta^2(\mathbf{x}; h_1, h_2) - (\mathrm{E}\,\beta(\mathbf{x}; h_1, h_2))^2 \tag{5.26}$$

As before, we expand as an *iid* sum and then omit the second term to get:

$$\sigma_b^2(\mathbf{x}; h_1, h_2) \approx \sum_{i=1}^{n} \left\{ \mathrm{E}\,\frac{\tilde{f}_i^2(\mathbf{x}; h_1)}{n^2} - 2\,\mathrm{E}\,\frac{\tilde{f}_i(\mathbf{x}; h_1)}{n}\,\frac{\tilde{f}_i(\mathbf{x}; h_2)}{n} + \mathrm{E}\,\frac{\tilde{f}_i^2(\mathbf{x}; h_2)}{n^2} \right\} \tag{5.27}$$

The first and last terms should look familiar; these are the same as the terms that we estimated earlier using Equation 5.10. The middle term is new, and is due to the fact that the two component estimates are partially correlated. As before, we create an estimator $\tilde{\tau}$, with this term as its expected value:

$$\tilde{\tau}(\mathbf{x}; h_1, h_2) = \frac{V^2}{n^2} \sum_{i=1}^{n} \mathrm{K}_{h_1}(\mathbf{x} - \mathbf{X}_i)\,\mathrm{K}_{h_2}(\mathbf{x} - \mathbf{X}_i)\,r^2(\lambda_i) \tag{5.28}$$

Our variance estimator for β is:

$$\tilde{\sigma}_\beta^2(\mathbf{x}; h_1, h_2) = \tilde{\sigma}^2(\mathbf{x}; h_1) + \tilde{\tau}(\mathbf{x}; h_1, h_2) + \tilde{\sigma}^2(\mathbf{x}; h_2) \tag{5.29}$$

Once we have an estimate of the standard deviation $\tilde{\sigma}_\beta$ for the estimator β we can decide if its value can be plausibly explained as noise alone or should be taken as evidence of bias. We choose a threshold C_β such that it is very unlikely that a random error will exceed $C_\beta\tilde{\sigma}_\beta$. If the value of $|\beta|$ does exceed this threshold, then we assume that there must have been a significant change in the bias.

It is important to choose C_β large enough that we only rarely mistake noise for bias, to avoid introducing distracting visual artifacts. Based on the normal distribution approximation, we typically use values in the range of 4 to 6 for C_β. An example of one step of bias detection is shown in Figure 5.4. Notice that the procedure largely succeeds in detecting the inadequately resolved shadows and, just as importantly, fails to detect bias in the surrounding areas of low bias.

We have only implemented the bias detection technique for our luminance estimates, but it could easily be applied to chromaticity estimates as well. We start with the

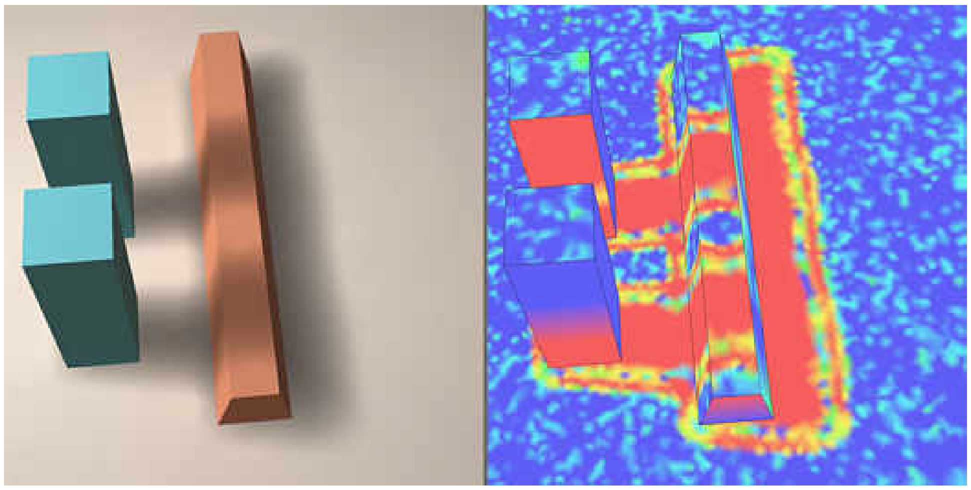

Figure 5.4: One level of bias detection. We start with the results (left) after choosing the bandwidths solely to eliminate visible noise. We then evaluate $|\beta/\tilde{\sigma}_\beta|$ (right, false color) using the initial bandwidths and bandwidths one step smaller. Blue, cyan, green, yellow, and red correspond to values of 1, 2, 4, 8, and 16 respectively. Blue/cyan indicate no detectable bias. Yellow/red indicate clearly detectable bias.

bandwidth determined using the Weber's law heuristic as our initial guess. Then we successively evaluate β using our current guess and a set of decreasing bandwidths. Currently we divide the bandwidth by $\sqrt[4]{2}$ at each step. At each bandwidth we evaluate the estimators given by Equations 5.5, 5.10, and 5.28. If we detect bias (i.e. $|\beta| \geq C_c\tilde{\sigma}_\beta$) at one of these smaller bandwidths, we substitute the estimate at that bandwidth as our new best guess and continue the process. As the bandwidth becomes too small, the noise level rises rapidly and the gaussian approximation for the error breaks down. Therefore, we stop the process whenever the standard deviation of an estimate exceeds some threshold (e.g., 7%).

So far we have estimated the bandwidth at each point independently, but it is sometimes desirable to enforce some smoothness of the bandwidths at nearby points. Because the bias can vary with the bandwidth, rapid fluctuations in the bandwidth can show up as visual artifacts in the results, especially when a larger bandwidth step size is used in the bias detection. We use a simple smoothing procedure based on the assumption that where bias is detected, there is likely to be bias at neighboring points, even if it was not detected. We use a rule that the bandwidth at neighboring points can only increase by 75% of their spatial separation and reduce bandwidths when necessary to enforce this.

We find that the bias detection tends to evaluate an average of between 6 and 12 bandwidths per vertex, but most of these are at smaller bandwidths which are less expensive to compute. On average, using bias detection increases the density estimation compute time by roughly a factor of two. We believe this is well justified by the improved quality of the results, such as shown in Figure 5.5, especially since density estimation often is only a small part of the total computation. In the density estimation framework algorithm, it accounts for only 10% to 15% of the computation with the particle tracing being by far the most expensive part.

Results and luminance bandwidths for two larger environments are shown in Figure 5.6. You can see how the bandwidths adapt automatically and are smaller in bright regions and near features. The chromaticity bandwidths are not shown because they are less interesting. Generally they are somewhat larger than the luminance bandwidths and do not decrease near features because we are not currently applying bias detection to them. All of the example figures in this chapter were computed using local constant density estimation (i.e. Equation 4.26).

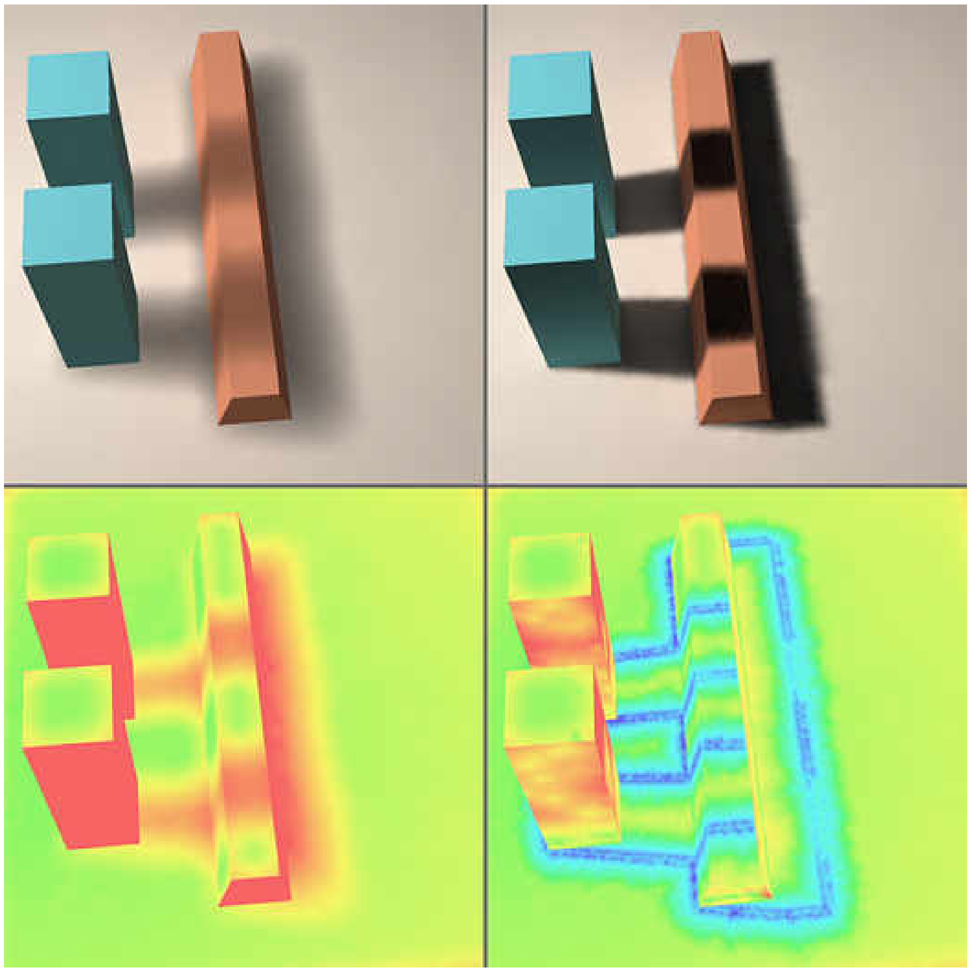

Figure 5.5: Results (top) before bias detection (left) and after bias detection (right). The luminance bandwidths are shown in false color (bottom) with blue, cyan, green, yellow, and red corresponding to bandwidths ranging from small to large. These results were generated using the same data as used in Figures 5.1 and 5.4.

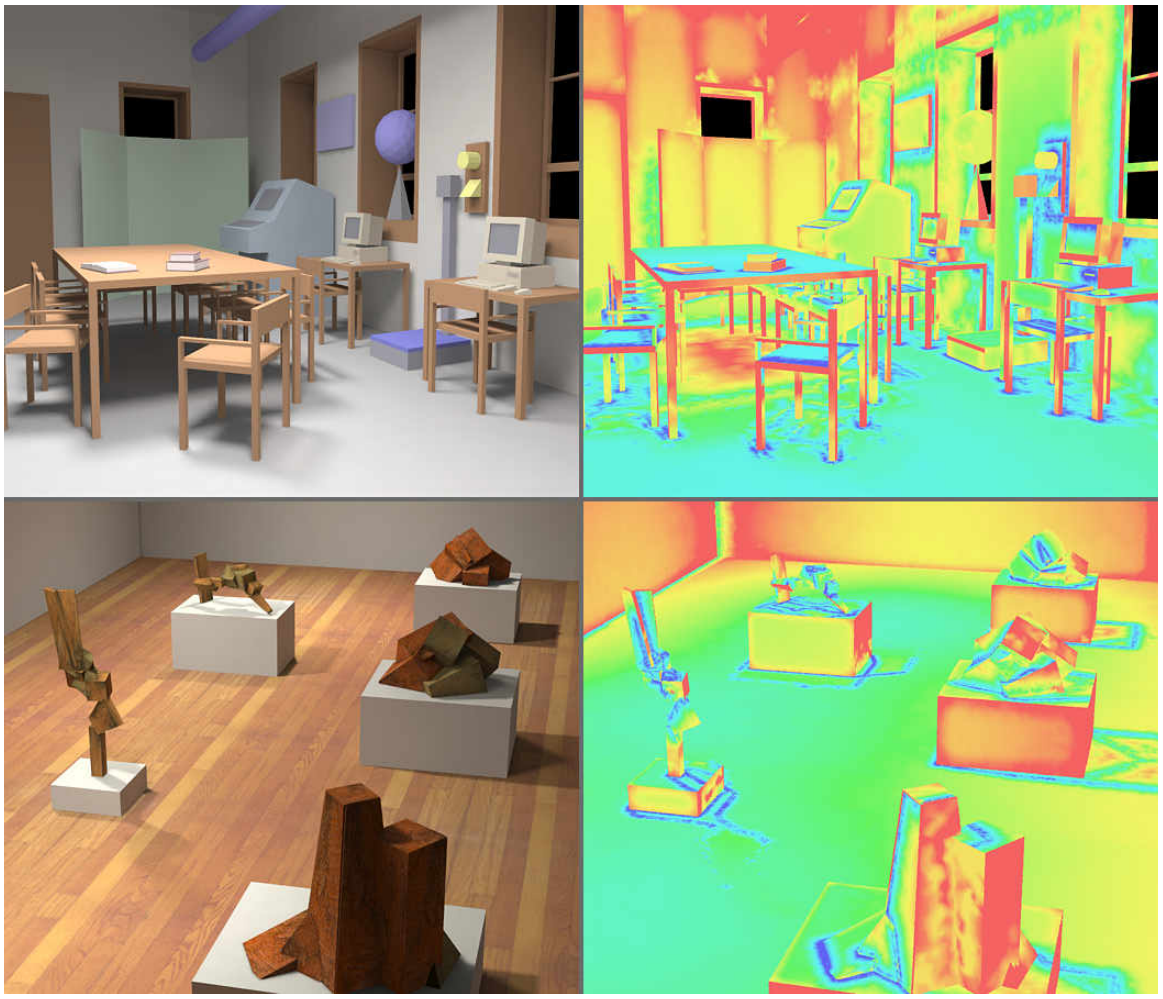

Figure 5.6: Images (left) of two environments and the luminance bandwidths (right) chosen by the bandwidth selector. Blue, cyan, green, yellow, and red correspond to bandwidths ranging from small to large. Notice that bandwidths are larger in dark unbiased regions and much smaller around feature boundaries.

Chapter 6

Further Issues and Optimizations

6.1 Importance Sampling

Importance sampling is a common technique used to increase the efficiency of Monte Carlo methods. The simplest way to improve solution quality is to use more samples (e.g., doubling the number of particles in our simulation), but this is costly in terms of computation time. Importance sampling techniques try reduce the number of samples needed by altering the ways the samples are generated to increase the average amount of useful information per sample. Since particle tracing is by far the most expensive part of our computations, techniques that can reduce the need for additional particles are very significant.

In the density estimation framework, importance sampling can be accomplished by modifying the particle emission and scattering probabilities (i.e. p_{e} and p_{s} from Section 3.2). Equivalently, we can vary the amount of power carried by individual particles since the particle probabilities and power mutually determine each other (Equations 3.1 and 3.2). In the previous chapters, we always constrained our particles to carry an equal amount of power, but we are free to vary the power between particles or even within the lifetime of a single particle. The challenge is choosing the power in ways that are beneficial and will actually improve our results.

The effective solution resolution that we achieve depends on the local bandwidths chosen in the density estimation. As discussed in Chapter 5, these bandwidths are chosen largely to meet certain variance, or noise, criteria. The variance in the density estimation for a particular bandwidth depends on the local density of particle hit points and the amount of variation in the weighting given to these points. Two importance sampling strategies are thus, to increase the homogeneity of the particle weights and to increase the local density of particle hits in important regions. The first strategy varies the particle power as a function of its non-spatial attributes (i.e. wavelength) and attempts to increase the effective resolution everywhere. The second is spatially dependent and trades increased resolution in some regions for decreased resolution in others.

6.1.1 Wavelength Importance Sampling

Our ability to perceive light is strongly dependent on its wavelength. Human vision is generally considered to be confined to the region between 380 to 800 nanometers, and our sensitivity varies considerably within this range. If we only need our results to match reality to a human observer, there is little benefit in wasting computation on wavelengths to which we have little or no sensitivity. In our initial implementations, we simulated only the range between 400 and 700nm which contains the vast majority of our visual sensitivity. Within this range each particle carried a constant amount of energy regardless of its wavelength.

Even within this limited range our sensitivity varies considerably. A better idea is to make the power carried by a particle depend on its wavelength and our visual sensitivity. We first choose an importance spectrum $\psi(\lambda)$ that specifies the relative importance of various wavelengths. During the particle tracing, we then ensure that the power ϕ associated with a particle is inversely proportional to its importance (i.e. $\phi(\lambda) \propto \frac{1}{\psi(\lambda)}$). This will automatically result in more particles at important wavelengths and fewer elsewhere. If the importance spectrum is well chosen, this will increase the overall efficiency of our simulation.

Of course we still need to choose a good importance spectrum. We want to increase the effective resolution by enabling the bandwidth selector to achieve its variance targets using smaller bandwidths. To formalize this goal, let us rewrite the relevant equations to handle a varying power associated with each particle hit. The basic density estimator (Equation 5.5) becomes:

$$\tilde{f}(\mathbf{x}; h) = \sum_{i=1}^{n} \phi_i \, \mathrm{K}_h(\mathbf{x} - \mathbf{X}_i) \, r(\lambda_i) \tag{6.1}$$

where ϕ_i is the power associated with the ith particle hit point. The expected value of this estimator is fixed by our requirement that the particle tracing be unbiased. Its expected value depends on the true function, the bandwidth, and the kernel, but not on our choice of an importance spectrum.

What does change when varying the particle power is the variance which we want to minimize. Allowing for varying power, our variance estimator (Equation 5.10) becomes:

$$\tilde{\sigma}^2(\mathbf{x}; h) = \sum_{i=1}^{n} \phi_i^2 \, \mathrm{K}_h^2(\mathbf{x} - \mathbf{X}_i) \, r^2(\lambda_i) \tag{6.2}$$

Note that this is simply the sum of the squares of the elements in the previous equation. If the number of elements and their sum is fixed, then a sum of squares is minimized when all the elements in the sum are equal[1]. This minimum would be achieved if we could use the spectral response function $r(\lambda)$ as our importance spectrum $\psi(\lambda)$. Unfortunately this is not generally possible.

We actually use many different spectral response functions $r(\lambda)$ even within a single solution, and our importance spectrum will have to be a compromise between these various functions. In our application, these spectral response functions are products of a local spectral reflectance function and of a trisimulus channel response function

[1]This is the same reason why the uniform kernel minimizes the variance among all kernels of fixed support and bandwidth.

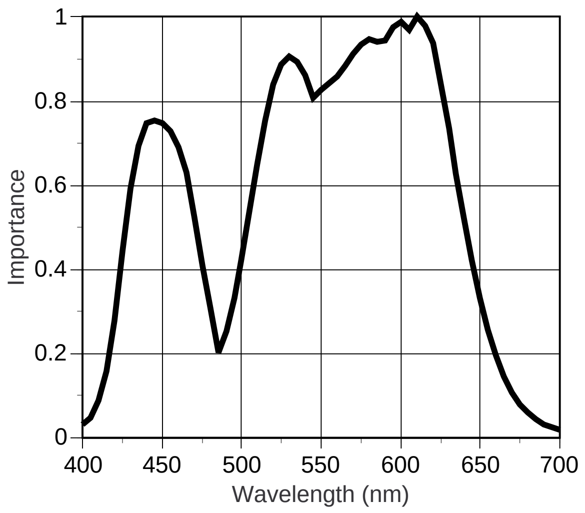

Figure 6.1: Empirical wavelength importance function. Having particles carry power inversely proportional to this importance function increases the efficiency of our computations. This spectrum was chosen based on human visual sensitivity information, measured typical spectral reflectances and some simple numerical experiments. The trimodal nature of this function is due to the three types of color receptors in human photopic vision.

(e.g., Equation 5.12). Different surface reflectances and the three tristimulus channels combine to produce many different composite response functions. The luminous channel is the most important part of the tristimulus values, but optimizing solely for it would cause excessive color noise. Similarly optimizing the importance for a particular surface's reflectance could cause excessive noise increases at other surfaces.

To find an importance spectrum that works reasonably well for generic scenes, we performed some simple numerical experiments using a set of measured spectral reflectance spectra[2]. We assumed the illumination had equal energy at all wavelengths (i.e. "flat" spectrum light) and used a noise measure based on the CIE L*a*b* color difference formula. A simulated annealing process was used to find the importance spectrum shown in

[2]These spectra are publicly available at ftp://ftp.eos.ncsu.edu/pub/spectra.

Table 6.1: Effect of wavelength-related noise on reconstructed spatial resolution. This table lists the number of luminance and chromiticity estimates for three different cases. In our system, these estimates are spaced relative to their respective bandwidths, and thus their number directly reflects the achieved spatial resolution. The sample scene consisted of a Macbeth ColorChecker [38] chart uniformly illuminated with uniform flat spectrum light or monochromatic light for the last column. In each trial, about four million particle hits were generated and local constant density estimation performed using uniform kernels and the bandwidth selector of Chapter 5 except that bias detection was not used. In the special case of monochromatic light, even a single particle provides complete chromaticity information and the potential chromaticity resolution is extremely high.

Vertices or Estimates	Uniform Power 400-700nm	Wavelength Importance Sampling	Monochromatic Light
# luminance	3983	5752	8123
relative efficiency	1	1.4	2
# chromaticity	990	1323	very large
relative efficiency	1	1.3	very large

Figure 6.1. While this importance spectrum is unlikely to be optimal for any particular scene, it should work well for a wide variety of scenes.

In order to estimate the effectiveness of our wavelength importance sampling we created a simple test scene consisting of a simulated Macbeth ColorChecker [38] chart. This chart is intended for color checking and calibration purposes in photography and consists of 24 squares of varying color with known spectral reflectances. Using uniform illumination, we simulated the chart both with and without wavelength importance sampling. We also simulated it illuminated with monochromatic light to represent the ideal case in which each particle carries complete spectral information and there is no wavelength-dependent noise. The results are summarized in Table 6.1.

Our simple experiment indicates that, compared to uniform particle power, a scheme that eliminated all spectral noise would approximately double the efficiency of our luminance reconstruction. Or in other words, we could achieve the same luminance resolution using only half as many particles. Using wavelength importance sampling gets us halfway to this ideal with approximately a $\sqrt{2}$ increase in efficiency. The potential gain in chromaticity resolution is much greater but perhaps less valuable than it seems. In practice, most chromaticity features are also luminance features and the luminance part is usually more perceptually important. Improving the luminance resolution is generally more important than improving the chromaticity resolution.

Several researchers have suggested that each particle could carry a complete spectrum instead of a single wavelength to better reduce the wavelength-related noise. Our simple experiment indicates that the potential gains in per-particle efficiency of such an approach over wavelength importance sampling seem to be fairly modest. Moreover, it would also introduce several implementational difficulties. Disk space is frequently one of our limiting factors and storing a complete spectrum with each particle would significantly increase the space needed per particle. For BSDFs whose spectral and angular parts are not separable, choosing the scattering probabilities would become significantly more difficult. In the worst cases, such as dispersion in glass, each particle will end up carrying a single wavelength anyway. In our opinion, the wavelength importance sampling approach gives of much of the potential wavelength-related efficiency gains with many fewer complications.

One of the advantages of wavelength importance sampling is that it can be implemented with only minimal changes to the basic particle tracer of Section 3.2. To ensure that each particle starts with a power inversely proportional to its importance ψ, we need only modify the emission probability slightly (compare to Equation 3.3).

$$p_{\mathrm{e}}(\mathbf{x},\omega,\lambda) \propto L_{\mathrm{emit}}(\mathbf{x},\omega,\lambda)\,\psi(\lambda)\,[\omega\cdot\hat{\mathbf{n}}] \tag{6.3}$$

Effectively all we need to do is multiply the emission spectrum of each light by the importance spectrum. The scattering probabilities are unchanged. Since we are already storing a wavelength with each particle hit, there is no need to store a power with each hit. A global table can be used to look up a hit's power based on its wavelength. The density estimation has to include each hit's power in its weighting, but since the weighting is already a function of wavelength (e.g., Equation 5.12), this is easily folded into the existing calculations.

In summary, wavelength importance allows us to gain a modest but noticable increase in solution efficiency at virtually no cost. However our experiments indicate that there is not much more benefit to be obtained from wavelength-based techniques. For further improvements in sampling, we need to look at spatially dependent techniques.

6.1.2 Spatially Varying Particle Power

When some regions are known to be more important or in greater need of resolution enhancement, it can be beneficial to use techniques that explicitly trade off improved resolution in some places for decreased resolution elsewhere. For example, in our framework as outlined so far, the density of particle hits is directly proportional to the illumination level. Hence the spatial resolution will be higher in bright regions and coarser in dark regions. Often it would be preferable to achieve a more uniform resolution everywhere.

We can achieve this by varying the average power associated with the hit points between regions. One way to accomplish this is to alter the directional components of the emission and scattering probabilities to preferentially send more particles toward important regions. Unfortunately, these probabilities have to be chosen before we know which surface the particle will actually strike next. At any point, we need some way of predicting which directions are most likely to result in the particle striking an important surface. In general, steering particles toward fairly broad targets is not difficult, but steering them with great precision toward many small targets is probably not feasible.

Another possibility is to take a split and prune approach. A higher power particle can be split into multiple lower power particles when it enters an important region. Each child particle is then traced independently from that point on. In unimportant regions, low power particles are turned into high power particles using a Russian roulette [4] algorithm. A major difficulty with splitting is that it tends to increase the correlation between particle hit points. All the particle hits derived from a single particle, even after splitting, are potentially correlated. If the correlation among nearby hit points is too large, it can invalidate the *iid* approximation that we used in our error analysis and bandwidth selection.

Due to these difficulties, we have not implemented a spatially varying importance sampling scheme, but we anticipate that they will be important tools in further enhancing the efficiency of the density estimation framework.

6.2 Use of Error Information

One of the advantages of the density estimation framework is that the errors in the solution process can be characterized relatively easily. Such error information is useful not only in determining the adequacy or inadequacy of a solution, but it can also be used as feedback to improve the solution process itself.

As discussed in Sections 2.4.4 and 4.1.2, our error is best analyzed by separating it into two components: systematic bias and random noise. While the particle tracing phase itself is unbiased, the raw particle data contains too much noise to be visualized directly. Thus we use the density estimation phase to reduce the noise at the cost of introducing some bias. Unlike other biased global illumination methods though, our bias is purely local in effect. This is the key in making our bias undertandable and controlled. In finite element methods by contrast, bias error introduced at one point is propagated globally in complex ways. By Equation 4.5, our bias is simply the difference between the true function and a smoothed, or blurred, version of the true function. The nature of this bias is easily understood although its actual numerical value is hard to estimate because it depends sensitively on details of the true function. A useful rule of thumb is that, in a particular solution, features larger than the local bandwidth are reliable while the presence or absence of features smaller than the local bandwidth is not.

The noise only depends on the blurred function and is thus much easier to estimate numerically (e.g., via Equation 5.10)[3]. Together the bandwidths and variance estimates provide us with more complete and accurate error information than is available in most other global illumination methods.

Such error information is potentially valuable for many purposes such as evaluating a resultant solution or judging the need for more accurate (and hence more expensive to compute) solutions. It can also be used to automatically choose the various parameters within the solution process. This is especially attractive as it helps to minimize the need for expertise or intervention on the part of the user. An excellent example of this approach is the bandwidth selector of Chapter 5. The error information can be even be used to trigger fallback methods outside of the framework if specified error targets are not being met.

6.2.1 The Small Polygon Problem

While we usually do not have pre-specified bandwidth targets, we do have specific variance targets based on the perceptual noise visibility thresholds in Chapter 5 and can easily detect when these noise targets are not being met. Kernel density estimation controls the variance by smoothing the data over some region. If total domain, which in our case is generally a polygon, is too small, even smoothing over the entire domain may not reduce the variance sufficiently. The result is visible noise on small polygons (Figure 6.2) with

[3]Similar empirical variance estimators can be used for any Monte Carlo method, but are not always reliable. The empirical variance is itself a Monte Carlo estimator with its own noise. This is a problem when a relatively unlikely event can have a disproportionately large effect on the estimate. Fortunately in our framework, the maximum contribution due to any single event (i.e. a particle hit point) is strictly limited and our variance estimates are generally reliable.

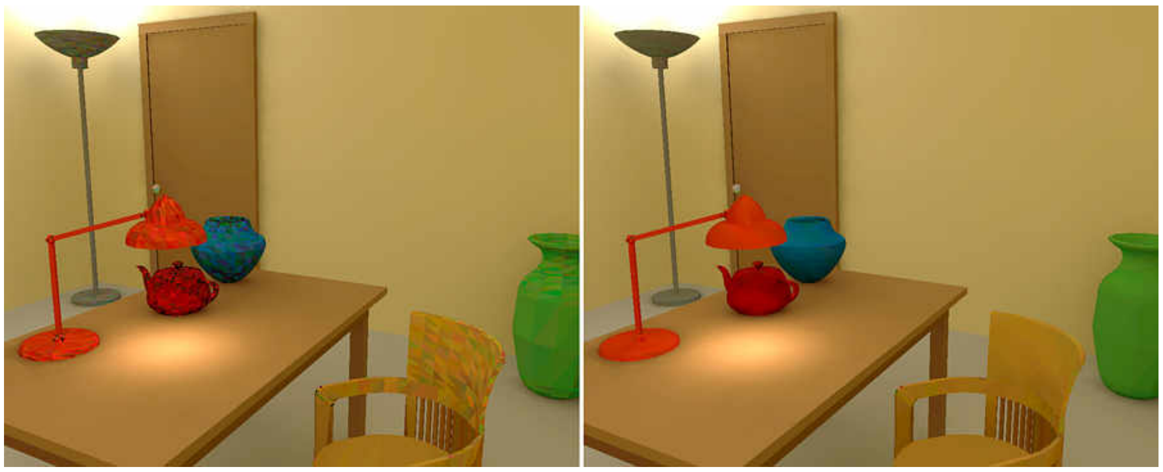

Figure 6.2: Small polygons in a scene may not contain enough particle hits for the density estimation to sufficiently reduce the local variance (left). One way to alleviate this problem is to share hit information between neighboring polygons that share vertices (right). The problem could be further reduced by sharing hit point information even more widely.

the color or chromatic aspect of noise typically being the most objectionable.

One way to reduce this problem is the increase the size of the density estimation domains and allow smoothing over multiple polygons. Nearby polygons and their hit points can be temporarily mapped into the local 2D space of the current polygon to form a larger domain. Smoothing over this larger region further reduces the variance, but may also increase the bias depending on similarity or disimilarity of the irradiance of the irradiance on the other polygons. Polygons which are similar in both location and orientation (e.g., polygons groups ones arising from polygonizing a curved surface[4]) tend to have the most similar illumination. Differing material properties among the surfaces is not a problem because we store incident particle information.

[4]In this case, an even better option would be working directly on the curved surface. Though we have not yet attempted this, we believe the local polynomial density estimation formulation of Chapter 4 provides a nice mathematical framework for such an extension.

If the polygons have significantly different orientations or surface normals, there are likely to be significant discontinuities in illumination between the polygons. Because of this, we may want to locally disable some density estimation features when sharing data between such polygons. For example, the local linear boundary bias correction method is only better than the local constant method when the function is locally smooth; thus we may prefer to use the local constant method in these cases. Also the bias detection algorithm in our bandwidth selector will sometimes detect and attempt to preserve these discontinuities. Since this would defeat the variance reduction we are trying to achieve, we generally disable the bias detection on these polygons.

The variance can also be reduced by introducing other types of bias than the usual smoothing bias of the density estimation. For instance, we could assume some default spectrum to use when there is a lack of better information. When the chromatic noise cannot be sufficiently reduced just by increasing the kernel size, light of the default spectrum can be added until the chromatic variance target is met. The result is a chromatic bias toward the hue of the default spectrum, but this may be less objectionable than the iridescent effect of visible chromatic noise.

6.3 Some Notes on Meshing

Computing the kernel density estimates is expensive and requires access to the voluminous particle data. It is not practical to evaluate them at display time. Instead we precompute the density estimates at a discrete set of vertices, and then interpolate values as needed during the interactive walkthroughs of our solutions. To keep our interpolation errors small, we want to ensure that the distance between vertices is sufficiently small. On the other hand, each vertex incurs computational, storage, and display costs, so we want to use as few as possible in meeting this goal.

In creating the solution mesh, we need to decide two issues: where to place the vertices and how to interpolate from them. For the latter, we use a triangulation and linear in-

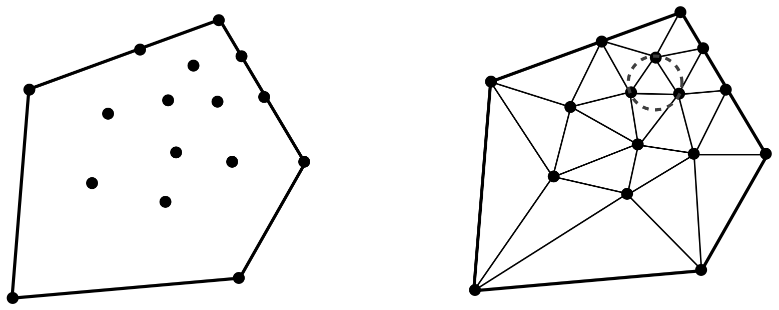

Figure 6.3: A polygon with some additional vertices (left) and its Delaunay triangulation (right). The circumcircle about any triangle (e.g., dashed circle) is guaranteed to not contain any other vertices.

terpolation within each triangle. This has the advantages of simplicity and rapid display using Gouraud shading which is supported by most 3D graphics display hardware. For any given set of vertices there are many different possible triangulations and, in general, we will not have enough information to choose the optimal one. Intuitively, we would like to always interpolate from the nearest set of surrounding vertices. We use Delaunay triangulations [19] because they have a property that roughly corresponds to this intuitive notion. They guarantee that the circumcircle about any triangle will not contain any other vertices in its interior (Figure 6.3). Technically we actually use constrained Delaunay triangulations because our triangulation must respect the boundaries of the original polygon. See [6] for a brief survey of some alternative possible meshing strategies.

To accurately capture a reconstructed feature, we need to ensure that the spacing between our vertices is smaller than the scale of the feature. In creating our mesh, we take advantage of the fact that the kernel density estimation process strongly blurs features that are smaller than the local bandwidth. Since only features at the bandwidth scale or larger will be accurately reconstructed anyway, we need only make our mesh spacing dense enough to capture features at these scales. We ensure this by putting constraints on the allowable edge lengths in our mesh based on the local bandwidth and adding vertices as necessary to enforce these constraints.

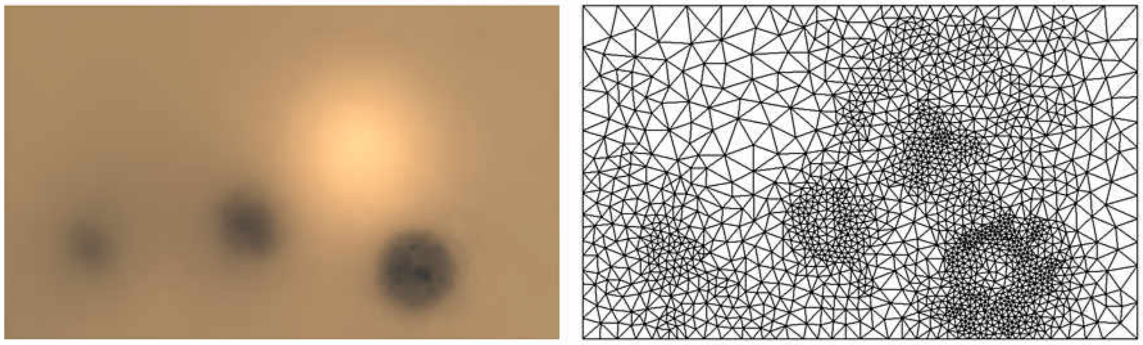

Figure 6.4: The table top from Figure 6.2 is shown here with both its estimated illumination function (left) and its underlying mesh (right). The mesh spacing (and bandwidth) are smaller in bright regions and near areas of rapid change.

Our initial implementation used a fixed bandwidth per surface. Thus we could construct the complete mesh before density estimation began. These meshes were generated using publicly available meshing code [46][5].

The adaptive bandwidth selector of Chapter 5, however, necessitated a more adaptive meshing approach. Currently we use a custom Delaunay triangulator. Once density estimation has been performed at a vertex and its local bandwidth chosen, we impose the constraint that no edge involving this vertex can be more than half its bandwidth in length. Vertices are added as needed to ensure that this constraint is both met and will continue to be met regardless of future vertex additions, and the triangulation is iteratively updated [35].

To try to minimize the total number of vertices added, we try to ensure that each new vertex is at least some minimum distance (e.g., 1/3 of the bandwidth) away from all existing vertices. Since the cost of constructing the mesh is neglible compared to the cost of performing the density estimation, it is worthwhile trying to optimize the placement

[5]Jonathan Shewchuk's *Triangle* meshing package is available at http://www.cs.cmu.edu/~quake/triangle.html which also provides some nice explanatory information about meshing and Delaunay triangulations.

of new vertices. The final result is an adaptive mesh that adapts to the bandwidth variations over a surface (e.g., Figure 6.4).

Note that we have been discussing meshing here specifically in relation to the density estimation phase. The decimation phase has somewhat different meshing requirements. In the density estimation phase the mesh needs to be dense enough to capture any detectable features that might exist. In the decimation phase, the mesh need only represent those features that were actually found, which is quite a different requirement. For example, the decimation phase performs significantly better if it is not restricted to Delaunay triangulations [56] and in some regions would benefit significantly from using higher order interpolation (e.g., [24]).

Chapter 7

Results

We have used the density estimation framework to generate solutions for a variety of environments. In this chapter, we will present some results using our current implementation. High quality solutions for four different models are used to show how performance is affected by model complexity. Next we demonstrate how the solution quality scales with computation time and the amount of particle data generated. Finally we show a comparision between measurements from a simple real environment and its simulation.

The computational components in our current implementation are:

- **Particle Tracing:** This program implements the governing equations given in Section 3.2. It is spectrally based with each particle carrying only a single wavelength of light. The power carried by each particle is determined using the wavelength importance spectrum shown in Figure 6.1. The particle tracing is the most expensive part of the computations. It is entirely computation bound with ray casting accounting for the majority of the time. We use a simple uniform spatial subdivision as our ray casting acceleration structure [3].

- **Sorting:** Our sorting code implements the two stage statistical sort described in Section 3.3. This part of the computation is largely I/O bound. It typically spends more than half its execution time waiting for particle information to be read from or written to disk.

- **Density Estimation:** This program implements the local polynomial kernel density estimation method described in Chapter 4. For these results, we use the local linear method and the uniform kernel. The bandwidths are chosen by the automatic selector described in Chapter 5 including the separation of luminance and chromaticity bandwidths and the use of our bias detection procedure to reduce the luminance bandwidths near illumination features. As discussed in Section 6.2.1, hit point information can be shared between neighboring polygons. Initially this is true for polygons who share a vertex and whose normal differ by less than 15 degrees. This limit is raised to 40 degrees if our noise targets are not met.

- **Decimation:** The results in this chapter do not include the decimation phase. A major focus of this thesis is the density estimation phase, and hence we have chosen to directly show the results it produces. Decimation is the fastest of our computational phases and has little effect on overall solution time. It is, however, essential when we want to display our solutions at interactive rates. Decimation performance results were previously reported in [56].

7.1 Performance and Sample Timings

The machine used for these timings is a 400 MHz Pentium II machine running Windows NT with 256 megabytes of main memory and two four gigabyte hard drives. Our implementation has been ported to several platforms and we frequently use parallel processing to speed up the solution process[1]. However for ease of comparison, all the reported tim-

[1]One of our design goals was to make it easy to get near linear parallel speedups at least for small numbers of processors. Our limited experience indicates that this is true although we have not fully parallelized our current implementation.

Table 7.1: Performance statistics for density estimation solutions of four environments. Listed is the original number of polygons in each environment, the time in hours to execute each phase and the number of triangles produced by the density estimation. No decimation phase was performed in these results. Each of these solutions used approximately 300 million particle hit points. Some images captured from these solutions are shown in Figures 7.1 to 7.4.

model		*Particle Tracing*	*Sorting*	*Density Estimation*		*Total*
Name	# Polygons	Time	Time	# Triangles	Time	Time
Science Center	3,780	5.3 hrs	0.8 hrs	1,705,582	1.6 hrs	7.6 hrs
Kidosaki	3,976	9.7 hrs	0.7 hrs	3,522,555	2.4 hrs	12.8 hrs
Library	13,729	18.6 hrs	0.7 hrs	3,441,944	2.6 hrs	21.9 hrs
Fallingwater	140,094	19.4 hrs	0.8 hrs	3,382,452	2.6 hrs	22.8 hrs

ings are for execution on a single processor. We configured our programs to use only about 100 megabytes of the memory and used one of the four gigabyte drives for temporary storage of particle data.

Timings for high quality solution of four architectural environments are shown in Table 7.1. Notice that although the complexity of the later models increased considerably, the solution times increased much more modestly. Approximately 300 million particle hit points were generated for each of these solutions. Solutions can be generated using more or less data as we will demonstrate in Section 7.1.1, but larger solutions would require more than the four gigabytes of disk space we allocated to hold temporary particle data. Images from these solutions are shown in Figures 7.1 to 7.4.

The models we used are:

- **Science Center:** This is a model of a computer room in the Science Center Museum in Ithaca, NY. Shown in Figure 7.1, it is the simplest of the four models and the fastest to solve. It is lit by two area luminaires near the ceiling, yet this is

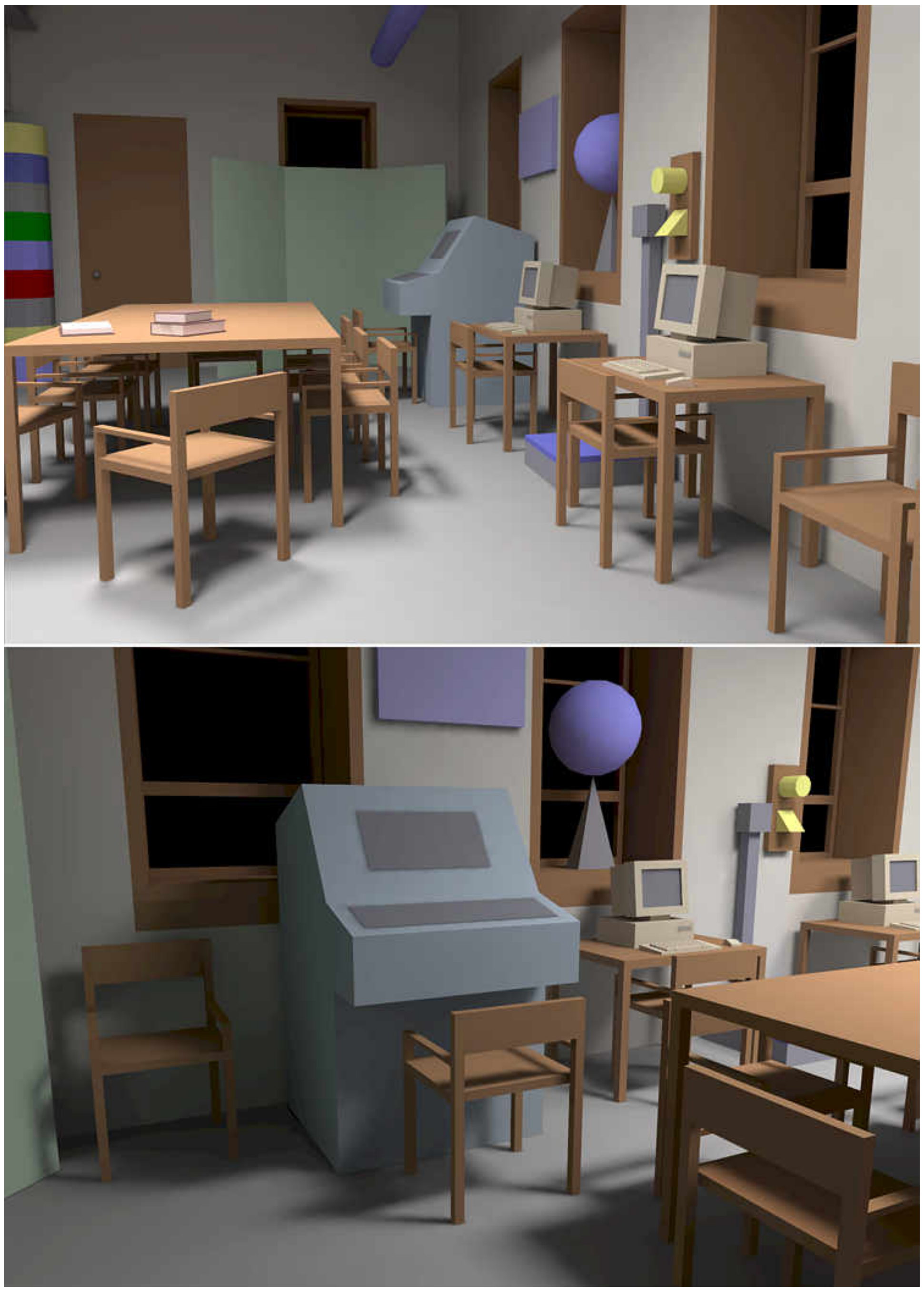

Figure 7.1: Two views of the Science Center solution listed in Table 7.1.

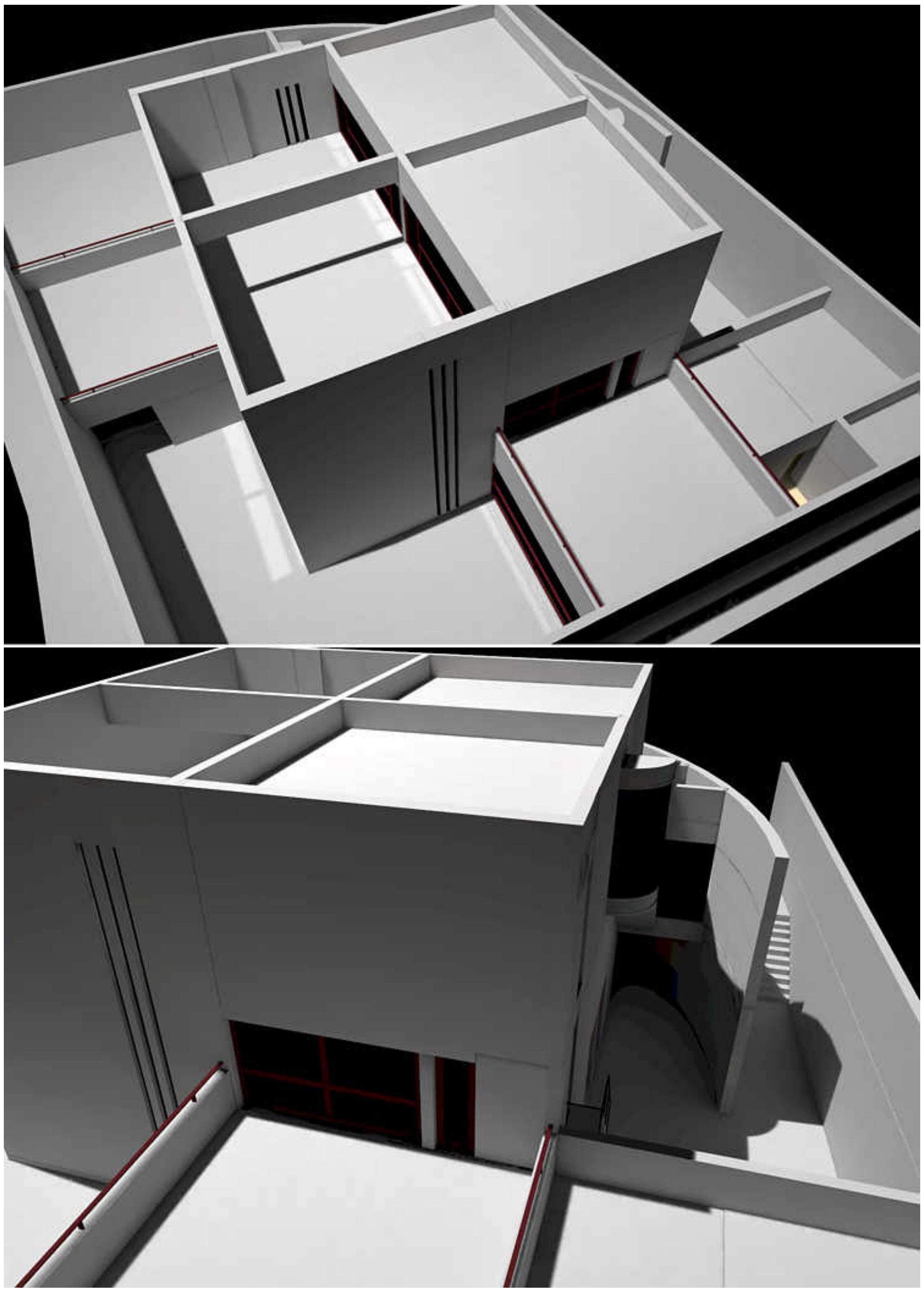

Figure 7.2: Two views of the Kidosaki solution listed in Table 7.1.

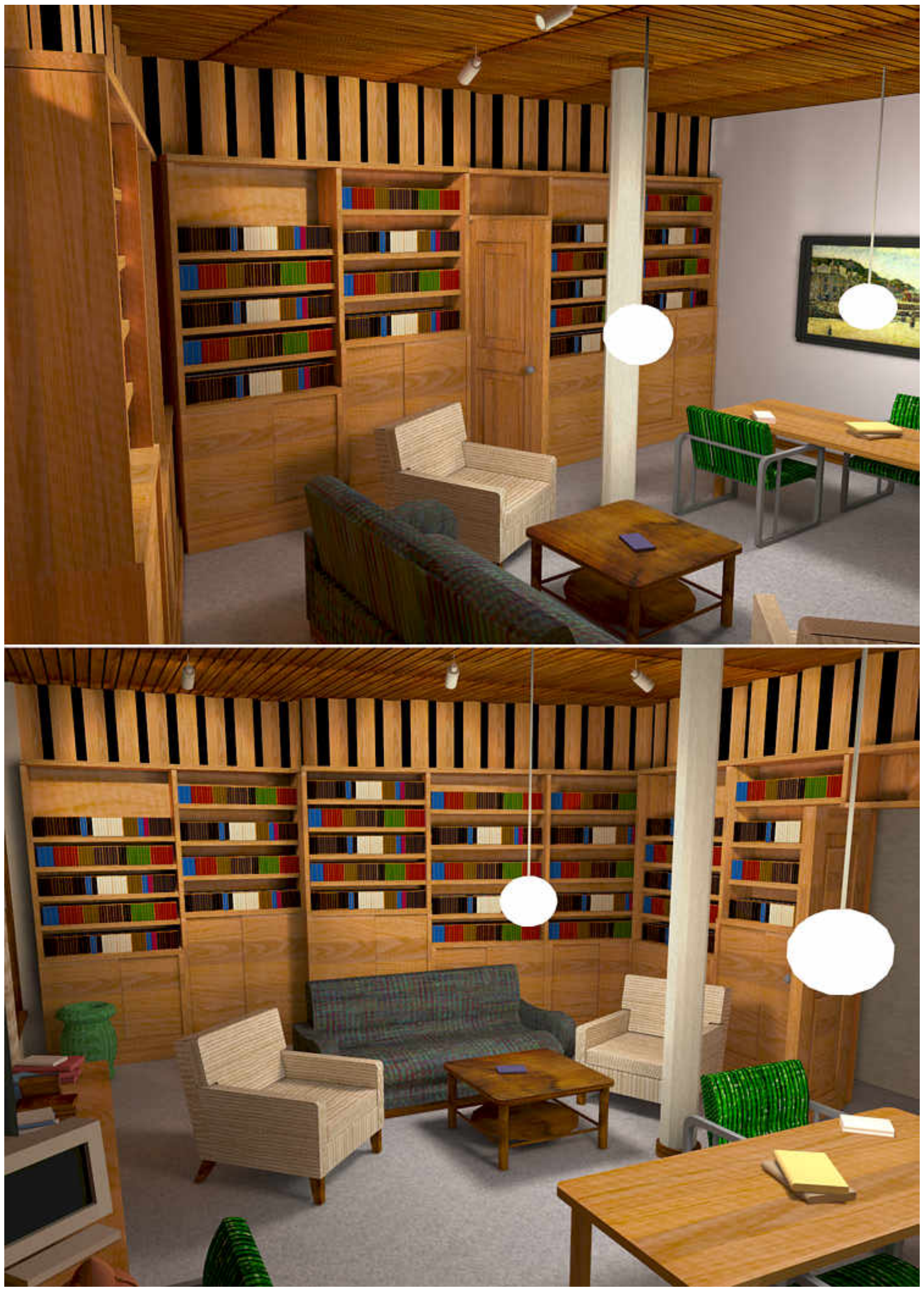

Figure 7.3: Two views of the Library solution listed in Table 7.1.

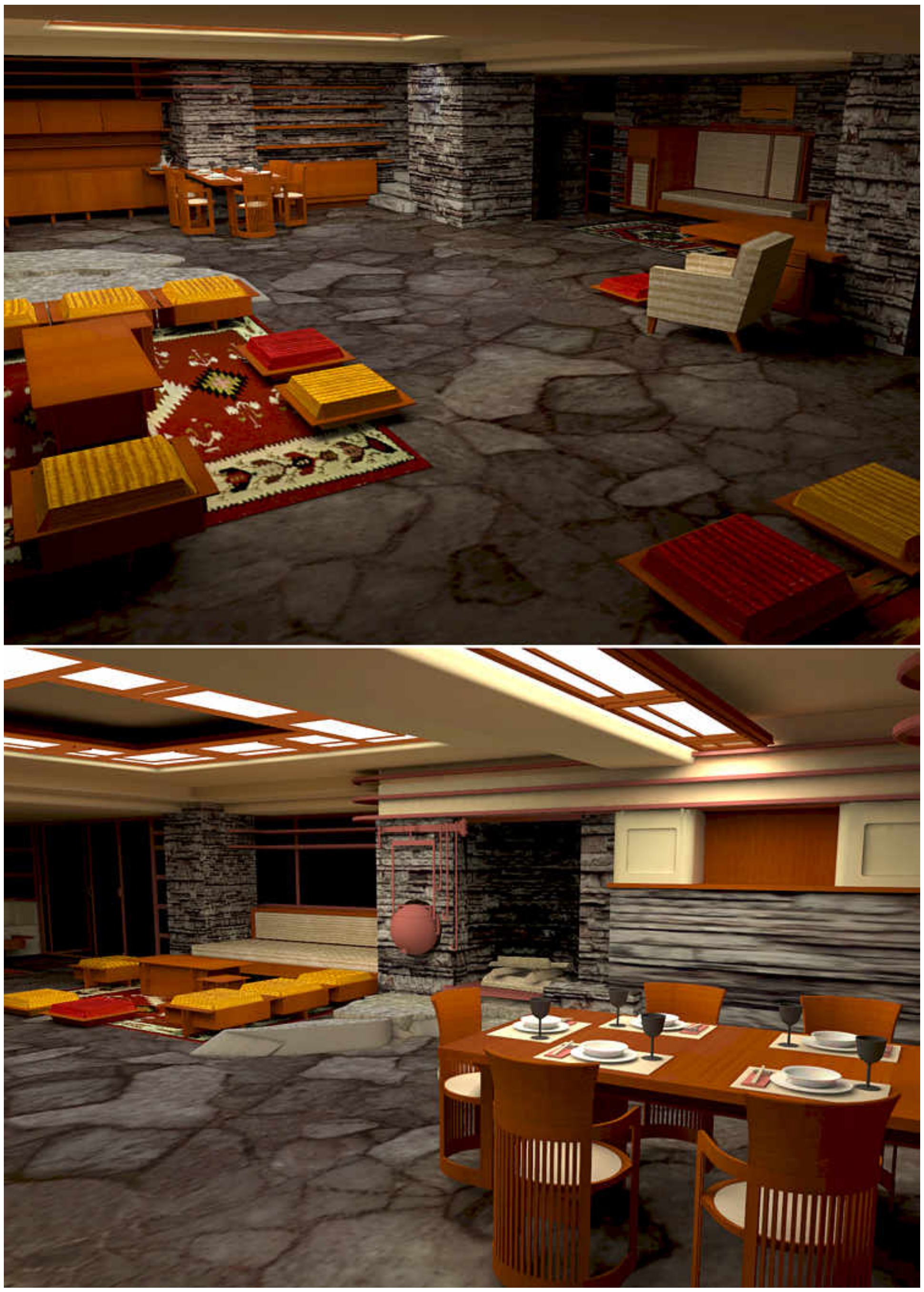

Figure 7.4: Two views of the Fallingwater solution listed in Table 7.1.

sufficient to create many interesting illumination details especially in the shadows on the floor.

- **Kidosaki:** Shown in Figure 7.2 is a model of the sunlit exterior of the Kidosaki house designed by Tadao Ando. The light is provided by a single distant spotlight. Unfortunately this is a somewhat inefficient way to simulate sunlight. A much better way would be to include an explicit sun/sky model in the particle tracer. One feature to notice is the caustics on several of the patios caused by light reflecting off the glass in the windows.

- **Library:** This is a model of the Villa Mairea Library which is a room in a private house designed by Alvar Aalto. This model is more geometrically complex, contains numerous texture maps, and more complex lighting as shown in Figure 7.3. It is lit by two suspended globe luminaires and several spotlights set near the ceiling. The black strips near the ceiling are actually mirrors. These images were captured as they would be displayed during an interactive walkthrough and without the more expensive display techniques needed to display non-diffuse surfaces correctly (See Figure 2.13). However, we do see their effects on other surfaces such as the caustic clearly visible in the upper left.

- **Fallingwater:** Figure 7.4 shows a model of one floor of the Fallingwater house designed by Frank Llyod Wright. This is a night time simulation with the light being provided by two large area luminaires set into the ceiling. Notice that while this is by far the most geometrically complex of our models, the computation time increased only modestly. This bolsters our claim that our framework scales well to complex environments.

Table 7.2: Times to produce solutions of Library model using varying amounts of particle data. Images of these solutions are shown in Figure 7.5.

Relative	*Particle Tracing*		*Sorting*	*Density Estimation*		*Total*
Quality	# Hit Points	Time	Time	# Triangles	Time	Time
$\frac{1}{64}$	4.7 million	0.29 hrs	0.01 hrs	173,474	0.08 hrs	0.38 hrs
$\frac{1}{16}$	18.8 million	1.16 hrs	0.04 hrs	442,277	0.23 hrs	1.42 hrs
$\frac{1}{4}$	75 million	4.64 hrs	0.16 hrs	1,185,667	0.79 hrs	5.59 hrs
1	300 million	18.59 hrs	0.71 hrs	3,441,944	2.60 hrs	21.90 hrs

7.1.1 Time vs. Quality Tradeoff

Depending on time and quality constraints, we can generate solutions using different amounts of particle data. Solutions using fewer particles can be generated more quickly but have decreased resolution of illumination features and more problems with noise on small surfaces. To illustrate this tradeoff we have generated some smaller solutions of the Library model from the previous section using a fourth, a sixteenth, and a sixty-fourth as much particle data. The timings in Table 7.2 show that execution time varies roughly linearly with the amount of particle data. Figure 7.5 shows how the quality varies with the amount particle data generated.

Note how the broad patterns of illumination are evident even in the coarsest solution. Details such as shadow boundaries smoothly refine as more particle data is used. If the user is aware of this behavior, then coarse solutions can be very valuable for rapid approximate feedback on the illumination in a model.

7.1.2 Scalability

One of the important characteristics of any global illumination algorithm is how well it scales to handle more complex environments. Unfortunately scaling is a rather compli-

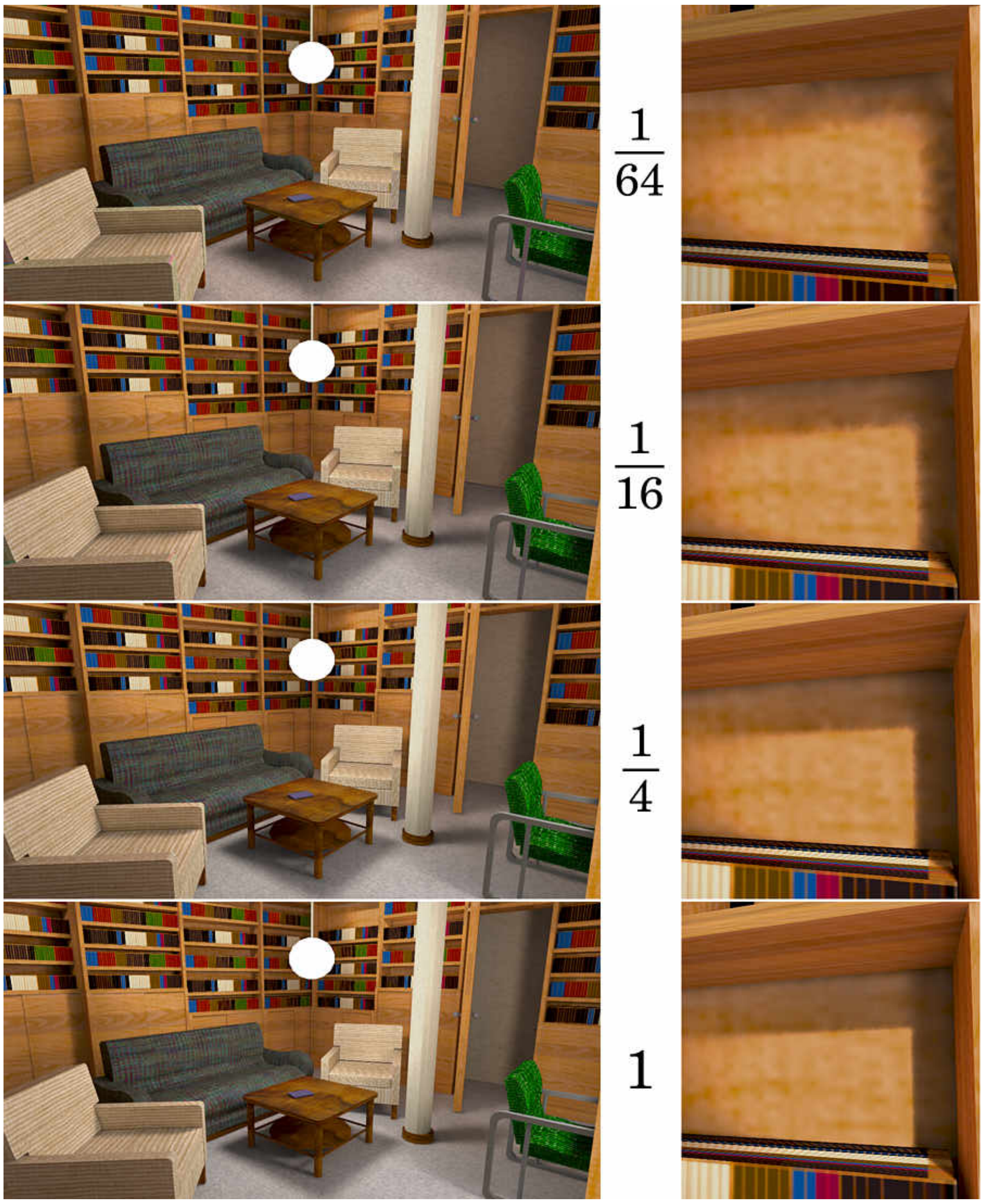

Figure 7.5: Solutions of the Libary model of varying quality as listed in Table 7.2. The right column shows a closeup of a shadow detail in one of the bookcases.

cated subject. There are many different ways in which models can become more complex and each is likely to have different effects on any particular algorithm. Below we list a few possibilities and discuss how they would individually affect our framework.

1. **Total surface area:** To achieve the same solution resolution, we would need to maintain the same average density of particle hit points. To achieve this, the amount of particle data generated would need to increase roughly linearly with the total surface area in the model. This should cause a roughly linear increase in the solution time.

2. **Number of surfaces:** Excluding small polygon problems, the amount of particle data needed does not depend on the number of surfaces. The particle tracing and density estimation times will increase somewhat due to visibility and initialization operations, but as indicated in Table 7.1, this is not a very strong effect.

3. **Average illumination level** This has no effect. The amount of power carried per particle simply scales up or down without a need for more particles.

4. **Illumination constrast** This is perhaps the most difficult case for the density estimation framework. Since particle density is directly related to the relative illumination intensity, increasing the illumination contrast will require many more particles to maintain a sufficient hit point density in the darker regions. Spatial importance sampling strategies will be very important future enhancements in reducing this problem.

5. **Material or scattering properties (BSDF):** Adding more complex scattering functions would have little effect, though particle tracing would become slightly more expensive due to the need to sample these more complex distributions. However, the irradiance and radiant exitance functions that we are computing become less useful as the BSDFs become less diffuse.

7.2 Comparison to a Measured Environment

In order to test the validity of our assumptions and our implementation, we would like to compare the results of one of our simulations against measurements from a real environment. Unfortunately this is a rather challenging and involved task. It requires precise measurements of both the input functions and a way to compare the results. Each of the relevant functions, geometry, scattering, emission, and radiance are typically complex and difficult to measure adequately.

Although the models in the previous sections were created based on real environments, they are inadequate for such a comparision. They are also greatly simplified as compared to the real environments. Many of their attributes, especially their material properties, were set "by eye" relying on the non-physical lighting models provided by the modeling system in which they were built. When comparing simulation results of such models, it is not possible to differentiate between errors due to erroneous input from simulation error.

To overcome this lack of calibrated input, the Cornell's Program of Computer Graphics created a simple environment known as the Cornell Box which was designed to be relatively easy to characterize and measure. The geometry consists of two cubes inside a larger cube. The walls were painted with flat latex paint which has a nearly Lambertian scattering function and their spectral reflectances were measured. The light source was designed to emit a nearly diffuse pattern of light and its spectrum was also measured. Images of the box were captured using a calibrated CCD camera and several narrow band spectral filters. The position of this camera was also measured so that simulated images could be generated from the same viewpoint and compared against the captured images. All of this data was made available at http://www.graphics.cornell.edu/online/box as a public service to allow researchers such as ourselves to check the quality of their rendering simulations.

Using this information, we simulated the lighting inside the Cornell Box and the comparison is shown in Figure 7.6. We believe that our simulated results match the real

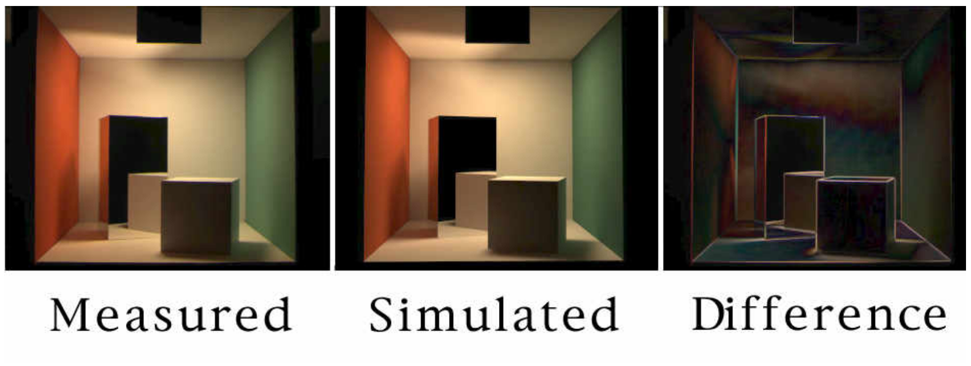

Figure 7.6: Measured and simulated images of the Cornell Box. Also shown is a difference image magnified by a factor of five.

results quite well and may be within the precision of the measured data in most places. The errors along object boundaries are due to some slight misalignment of the geometry. Some error is also visible along some shadow boundaries due to inadequate resolution in our simulation. The differences on the upper walls are likely due to some variation between the true and measured emission functions at oblique angles. Aside from these errors, our results agree extremely well with the measured images.

We did need to make one change to the framework as described to make this comparison. Normally we transform from spectral to XYZ space to simulate the response of a human observer, but in this case we needed to simulate the response of the camera instead. We simulated the response of the camera with each of the three filters individually and then composited the resulting three grayscale images to form the color image shown. To produce the grayscale images, we eliminated the chromatic estimation step and using the combined camera and filter reponse curves instead of the luminous efficiency curve we normally use. There is also an overall scaling factor in the measured data that is not known very precisely, hence each image was scaled to match the average intensity of the measured images.

Chapter 8

Conclusion

8.1 Summary

We have presented a new framework for producing view-independent global illumination solutions. There are three principal contributions in this work: the framework's separation of the transport and reconstruction computations, its ability to produce accurate solutions with precisely known error charateristics, and the techniques we have developed to improve its accuracy and efficiency.

The global illumination problem is usually formulated as a balance equation (e.g., Equation 2.28) in terms of the spectral radiance. Most previous approaches have tried to explicitly solve this equation. Unfortunately in realistic complex environments, the spectral radiance is an extremely complicated and detailed function. Except in special cases it is not practical to assume that we can find or represent the spectral radiance very accurately. Thus view-independent methods typically solve for some simplified illumination function such as the radiant exitance. Explicitly solving the governing equation would in general require us to first solve for the complete spectral radiance, a computationally difficult if not impossible task.

Particle tracing, on the other hand, is capable of simulating the light transport without ever explicitly solving for the spectral radiance. Moreover it is unbiased and uses a relatively complete physical model of light transport for increased accuracy. Although the conventional wisdom has been that particle tracing is simply too expensive to be used exclusively to handle all light transport in a large environment, our work has shown that this is not the case.

We begin by separating the global illumination simulation into separate stages for the computation of the global light transport via particle tracing and the reconstruction of local illumination functions via density estimation. Previous researchers have avoided this approach because of the large amount of particle data that must be stored, but we have shown that this limitation can be overcome and that our procedure has many advantages. By separating global and local computations, we reduce the computational complexity of each stage, improve our scalability to handle complex environments, and expose abundant easily exploited parallelism. Furthermore delaying the density estimation decisions until all particle data is available allows us to use a non-parametric and data-driven approach to extract better solutions from the expensive particle data.

Particle tracing's generality also allows us to eliminate or delay most of our simplifying assumptions, increases our accuracy and streamlines the error analysis. The density estimation then directly estimates the desired simplified functions from the particle data. It does introduce some bias and simplifying assumptions, but it does so in a controlled and purely local fashion that makes its error characteristics easily understood. Such precise error information is unique among current view-independent global illumination techniques.

The density estimation phase recovers radiant exitance values and stores them at the vertices of a conservatively dense mesh. We introduce a further decimation phase to optimize this mesh for compactness and speed which allows us to display our solutions at interactive frame rates. The major features of the framework are summarized in Table 8.1.

Particle tracing is by far the most expensive part of our computations. In principle,

Table 8.1: Benefits of the density estimation framework.

Features	*Advantages*
Separation of transport and representation computations	Reduces computational complexity Simulates the full spectral radiance function. Reconstructs simplified illumination functions. Error is easily characterized. Easy to parallelize.
Particle tracing phase	Simulates general physical model of light transport. Scales well with model complexity.
Density estimation phase	Allows a non-parametric, data-driven approach to extract more information from the particle data.
Decimation phase	Optimizes mesh for speed and compactness.

even a straight-forward, naively implemented version of the density estimation framework would be capable of producing accurate solutions; however the number of particles required would be prohibitively large. To make high quality solutions feasible, we have introduced a number of techniques to improve the accuracy and efficiency of the framework and reduce the number of particles required. The most important of these are the components of our adaptive bandwidth selector: separation of luminance and chromaticity bandwidth, perceptually-motivated noise visibility predictors, and our statistical bias detection procedure. These techniques are listed in Table 8.2.

Using these techniques we have shown that we can generate high quality solutions of complex environments. We have also validated our simulation by comparing it to measured data for a real environment. The framework can be used to produce quick coarse solutions, but the real contribution of this work is the ability to produce accurate solutions with precisely known error charateristics. The density estimation framework can simulate a wider variety of lighting effects, with fewer simplifying assumptions, and a more precise error analysis than current view-independent global illumination methods.

There are many potential applications for such high quality solutions. One is as bench-

Table 8.2: Accuracy and efficiency enhancements.

Enhancements	*Advantages*
Separation of luminance and chromaticity bandwidths	Separately optimizes their bias vs. noise tradeoffs. Reduces color noise while sharpening luminance.
Perceptually-based noise visibility predictors	Finds bandwidths that eliminate visual noise. Sets upper bound on range of useful bandwidths.
Statistical bias detection	Finds underresolved illumination features. Spatially adapts bandwidths to reveal such features. Independent of the underlying physical cause.
Local polynomial density estimation techniques	Reduces or eliminates boundary bias. Implemented for arbitrary polygonal domains.
Wavelength importance sampling	Reduces wavelength related noise in particle data. Increases average information content per hit point.
Hit point sharing	Reduces noise problem on small polygons.
Adaptive mesh generation	Optimizes placement of density estimation locations. Reduces the number of kernel evaluations required.

mark, or reference solution, for judging the quality and effectiveness of more approximate but faster rendering methods. Another is in enabling applications where predictive results are required, for example, in helping a lighting engineer evaluate potential workspace designs.

8.2 Future Work

Although we have greatly improved the efficiency of our simulations, there are still many possibilities for further improvements. The current implementation is restricted to kernels with circular supports, but elliptical kernels can be more spatially adaptive and effective. They would require a better bandwidth selector to choose the required additional bandwidth parameters. Bias detection procedures specifically tuned to detect boundaries such as shadow or caustic edges could better resolve such features. The local polynomial density estimation method could be extended to work directly with curved surfaces to reduce the small polygon problems.

A wide variation in illumination levels can be problematic for the current implementation such as when simultaneously simulating both the sunlit exterior of a building and its dimly lit interior. In this case some form of guidance such as a global importance function could ensure that enough particles reach all locations.

Another limitation is that we currently only reconstruct directionless quantities such as radiant exitance. These only provide complete lighting information at diffuse surfaces. It is possible to reconstruct functions that capture directional as well as spatial variations of the radiance on the non-diffuse surfaces, but it will necessitate a tradeoff between spatial and directional resolutions. New methods would be needed to choose this tradeoff intelligently.

While our simulations can be highly accurate, they can only be as accurate as the input provided. Exploring how input errors in the input affect the simulation would provide valuable information for researchers and designers.

As presented the density estimation framework is already a state of the art system for producing accurate view-independent solutions, and with future enhancements we expect it to become an important tool in many global illumination applications.

Bibliography

[1] A. Appel. Some techniques for shading machine renderings of solids. In *AFIPS 1968 Spring Joint Computing Conference*, pages 37–49, 1968.

[2] James Arvo. Backward ray tracing. *Developments in Ray Tracing*, pages 259–263, 1986. ACM Siggraph '86 Course Notes.

[3] James Arvo and David Kirk. A survey of ray tracing acceleration techniques. In Andrew S. Glassner, editor, *An Introduction to Ray Tracing*. Academic Press, San Diego, CA, 1989.

[4] James Arvo and David Kirk. Particle transport and image synthesis. *Computer Graphics*, 24(3):63–66, August 1990. ACM Siggraph '90 Conference Proceedings.

[5] James Richard Arvo. *Analytic Methods For Simulated Light Transport*. PhD thesis, Yale, December 1995.

[6] Marchall Bern and David Eppstein. Mesh generation and optimal triangulation. In Ding-Zhu Du and Frank Hwang, editors, *Computing in Euclidean Geometry*, volume 4 of *Lecture Notes Series on Computing*, pages 47–123. World Scientific, Singapore, 1995.

[7] Patrick Billingsley. *Probability and Measure*. Wiley, New York, 1995.

[8] H. Richard Blackwell. Luminance difference thresholds. In *Handbook of Sensory Physiology*, volume VII/4: Visual Psychophysics, pages 78–101. Springer-Verlag, 1972.

[9] Mark R. Bolin and Gary W. Meyer. A perceptually based adaptive sampling algorithm. *Computer Graphics (ACM Siggraph '98 Conference Proceedings)*, pages 299–310, July 1998.

[10] Shenchang Eric Chen, Holly Rushmeier, Gavin Miller, and Douglass Turner. A progressive multi-pass method for global illumination. *Computer Graphics*, 25(4):165–174, July 1991. ACM Siggraph '91 Conference Proceedings.

[11] CIE. Industrial color-difference evaluation. CIE Pub. 116, Commission International de L'Eclairage, Vienna, 1995.

[12] Steven Collins. Adaptive splatting for specular to diffuse light transport. In *Proceedings of the Fifth Eurographics Workshop on Rendering*, pages 119–135, June 1994.

[13] Robert L. Cook and Kennneth E. Torrance. A reflectance model for computer graphics. *Computer Graphics*, 15(3):307–316, August 1981. ACM Siggraph '81 Conference Proceedings.

[14] D. A. Danielson. *Vectors and Tensors in Engineering and Physics*. Addison-Wesley, New York, 1991.

[15] Mark D. Fairchild. *Color Appearance Models*. Addison-Wesley, Reading, Massachusetts, 1998.

[16] J. Fan and I. Gijbels. *Local Polynomial Modeling and Its Applications.* Chapman and Hall, London, 1996.

[17] E. A. Feibush, M. Levoy, and R. L. Cook. Synthetic texturing using digital filters. *Computer Graphics (SIGGRAPH '80 Proceedings)*, pages 294–301, July 1980.

[18] James A. Ferwerda, Sumanta N. Pattanaik, Peter Shirley, and Donald P. Greenberg. A model of visual masking for computer graphics. *Computer Graphics (ACM Siggraph '97 Conference Proceedings)*, pages 143–152, August 1997.

[19] Steven Fortune. Voronoi diagrams and delaunay triangulations. In Ding-Zhu Du and Frank Hwang, editors, *Computing in Euclidean Geometry*, volume 4 of *Lecture Notes Series on Computing*, pages 225–265. World Scientific, Singapore, 1995.

[20] Andrew S. Glassner. *Principles of Digital Image Synthesis.* Morgan-Kaufman, San Francisco, 1995.

[21] Steven J. Gortler, Peter Schröder, Michael F. Cohen, and Pat Hanrahan. Wavelet radiosity. *Computer Graphics*, pages 221–230, August 1993. ACM Siggraph '93 Conference Proceedings.

[22] Donald P. Greenberg, Kenneth E. Torrance, Peter Shirley, James Arvo, James A. Ferwerda, Sumanta Pattanaik, Eric Lafortune, Bruce Walter, Sing-Choong Foo, and Ben Trumbore. A framework for realistic image synthesis. *Computer Graphics (ACM Siggraph '97 Conference Proceedings)*, pages 477–494, August 1997.

[23] Pat Hanrahan, David Salzman, and Larry Aupperle. A rapid hierarchical radiosity algorithm. *Computer Graphics*, 25(4):197–206, July 1991. ACM Siggraph '91 Conference Proceedings.

[24] Steven Hardt and Seth Teller. High-fidelity radiosity rendering at interactive rates. In *Rendering Techniques '96*, pages 71–80. Springer-Verlag/Wien, 1996.

[25] T. Hastie and C. Loader. Local regression: Automatic kernel carpentry. *Statist. Science*, 8:120–143, 1993.

[26] Xiao D. He, Kenneth E. Torrence, François X. Sillion, and Donald P. Greenberg. A comprehensive physical model for light reflection. *Computer Graphics*, 25(4):175–186, July 1991. ACM Siggraph '91 Conference Proceedings.

[27] Paul S. Heckbert. Adaptive radiosity textures for bidirectional ray tracing. *Computer Graphics*, 24(3):145–154, August 1990. ACM Siggraph '90 Conference Proceedings.

[28] Leo M. Hurvich. *Color Vision.* Sinauer Associates Inc., Sunderland, Massachusetts, 1981.

[29] David S. Immel, Michael F. Cohen, and Donald P. Greenberg. A radiosity method for non-diffuse environments. *Computer Graphics*, 20(4):133–142, August 1986. ACM Siggraph '86 Conference Proceedings.

[30] American National Standards Institute. ANSI standard nomenclature and definitions for illuminating engineering,. ANSI/IES RP-16-1986, Illuminating Engineering Society, 345 East 47th Street, New York, NY 10017, June 1986.

[31] Henrik Wann Jensen. Global illumination using photon maps. In *Rendering Techniques '96*, pages 21–30. Springer-Verlag/Wien, 1996.

[32] M. C. Jones, J. S. Marron, and S. J. Sheather. A brief survey of bandwidth selection for density estimation. *J. of the Amer. Statist. Assoc.*, 91:401–407, 1996.

[33] James T. Kajiya. The rendering equation. *Computer Graphics*, 20(4):143–150, August 1986. ACM Siggraph '86 Conference Proceedings.

[34] Eric P. F. Lafortune, Sing-Choong Foo, Kenneth E. Torrance, and Donald P. Greenberg. Non-linear approximation of reflectance functions. *Computer Graphics (ACM Siggraph '97 Conference Proceedings)*, pages 117–126, August 1997.

[35] Dani Lischinski. Incremental delaunay triangulation. In Paul Heckbert, editor, *Graphics Gems IV*, pages 47–59. Academic Press, Boston, 1994.

[36] Dani Lischinski, Filippo Tampieri, and Donald P. Greenberg. Combining hierarchical radiosity and discontinuity meshing. *Computer Graphics*, pages 199–208, August 1993. ACM Siggraph '93 Conference Proceedings.

[37] Jerry B. Marion and Mark A. Heald. *Classical Electromagnetic Radiation.* Academic Press College, Orlando, 1980.

[38] C. S. McCamy, H. Marcus, and J. G. Davidson. A color-rendition chart. *Journal of Applied Photographic Engineering*, 2(3):95–99, 1976.

[39] Karol Myszkowski. Lighting reconstruction using fast and adaptive density estimation techniques. In *Rendering Techniques '97*, pages 251–262. Springer-Verlag/Wien, 1997.

[40] Fred E. Nicodemus. Radiance. *Am. J. Phys*, 31:368–377, 1963.

[41] Michael Oren and Shree K. Nayar. Generalization of lambert's reflectance model. In Andrew Glassner, editor, *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series, pages 239–246. ACM SIGGRAPH, ACM Press, July 1994.

[42] S. N. Pattanaik. *Computational Methods for Global Illumination and Visualisation of Complex 3D Environments.* PhD thesis, Birla Institute of Technology & Science, Computer Science Department, Pilani, India, February 1993.

[43] Allen B. Poirson and Brian A. Wandell. Pattern-color separable pathways predict sensitivity to simple colored patterns. *Vision Research*, 36:515–526, 1996.

[44] R.A. Redner, M.E. Lee, and S.P. Uselton. Smooth b-spline illumination maps for bidirectional ray tracing. *ACM Transactions on Graphics*, 14(4), October 1995.

[45] Holly Rushmeier, Charles Patterson, and Aravindan Veerasamy. Geometric simplification for indirect illumination calculations. In *Proceedings of Graphics Interface '93*, pages 227–236, Toronto, Ontario, Canada, May 1993. Canadian Information Processing Society.

[46] Jonathan Richard Shewchuk. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In Ming C. Lin and Dinesh Manocha, editors, *Applied Computational Geometry: Towards Geometric Engineering*, volume 1148 of *Lecture Notes in Computer Science*, pages 203–222. Springer-Verlag, May 1996. From the First ACM Workshop on Applied Computational Geometry.

[47] Peter Shirley, Bretton Wade, Philip M. Hubbard, David Zareski, Bruce Walter, and Donald P. Greenberg. Global illumination via density-estimation. In *Rendering Techniques '95*, pages 219–230. Springer-Verlag/Wien, 1995.

[48] Peter Shirley, Bretton Wade, David Zareski, Philip Hubbard, Bruce Walter, and Donald P. Greenberg. Global illumination via density estimation. In *Proceedings of the Sixth Eurographics Workshop on Rendering*, pages 187–199, June 1995.

[49] B. W. Silverman. *Density Estimation for Statistics and Data Analysis.* Chapman and Hall, London and New York, 1986.

[50] Brian E. Smits, James R. Arvo, and Donald P. Greenberg. A clustering algorithm for radiosity in complex environments. *Computer Graphics*, 28(3):435–442, July 1994. ACM Siggraph '94 Conference Proceedings.

[51] Wolfgang Stürzlinger and Rui Bastos. Interactive rendering of globally illuminated glossy scenes. In *Rendering Techniques '97*, pages 93–102. Springer-Verlag/Wien, 1997.

[52] Seth Teller, Celeste Fowler, Thomas Funkhouser, and Pat Hanrahan. Partitioning and ordering large radiosity calculations. *Computer Graphics*, 28(3):443–450, July 1994. ACM Siggraph '94 Conference Proceedings.

[53] Eric Veach. Non-symmetric scattering in light transport algorithms. In *Rendering Techniques '96*, pages 81–90. Springer-Verlag/Wien, 1996.

[54] Eric Veach and Leonidas J. Guibas. Metropolis light transport. *Computer Graphics (ACM Siggraph '97 Conference Proceedings)*, pages 65–76, August 1997.

[55] Bretton Spencer Wade. Kernel based density estimation for global illumination. Master's thesis, Program of Computer Graphics, Cornell University, January 1996.

[56] Bruce Walter, Philip M. Hubbard, Peter Shirley, and Donald P. Greenberg. Global illumination using local linear density estimation. *ACM Transactions on Graphics*, July 1997.

[57] M. P. Wand and M. C. Jones. *Kernel Smoothing.* Chapman and Hall, London and New York, 1995.

[58] Gregory J. Ward. Measuring and modeling anisotropic reflection. *Computer Graphics*, 26(4):265–272, July 1992. ACM Siggraph '92 Conference Proceedings.

[59] Stephen H. Westin, James R. Arvo, and Kenneth E. Torrance. Measuring and modeling anisotropic reflection. *Computer Graphics*, 26(2):255–264, July 1992. ACM Siggraph '92 Conference Proceedings.

[60] Steven Wolfram. *Mathematica: a system for doing mathematics by computer.* Addison-Wesley, New York, 1991.

[61] Günter Wyszecki and W.S. Stiles. *Color Science: Concepts and Methods, Quantitative Data and Formulae.* Wiley, New York, N.Y., 1982.

[62] David Zareski, Bretton Wade, Philip Hubbard, and Peter Shirley. Efficient parallel global illumination using density estimation. In *Proceedings of the 1995 Parallel Rendering Symposium*, 1995.